PM-1 3 1/2"x4" PHOTOGRAPHIC MICROCOPY TARGET
NBS 1010a ANSI/ISO #2 EQUIVALENT

PRECISIONSM RESOLUTION TARGETS

National Library
of Canada

Acquisitions and
Bibliographic Services Branch

395 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale
du Canada

Direction des acquisitions et
des services bibliographiques

395, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file Votre référence

Our file Notre référence

NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

Canada

**A CLASSIFIER SYSTEM-BASED MODEL
OF SOME CHARACTERISTIC FEATURES OF CONSCIOUSNESS**

by

Christopher David Viger, BSc, MSc

**A thesis submitted to
the Faculty of Graduate Studies and Research
in partial fulfilment of
the requirements for the degree of**

Master of Arts

Department of Philosophy

Carleton University

Ottawa, Ontario

August 24, 1994

© copyright

1994, Christopher D Viger

National Library
of Canada

Acquisitions and
Bibliographic Services Branch

385 Wellington Street
Ottawa, Ontario
K1A 0N4

Bibliothèque nationale
du Canada

Direction des acquisitions et
des services bibliographiques

385, rue Wellington
Ottawa (Ontario)
K1A 0N4

Your file Votre référence

Our file Notre référence

**THE AUTHOR HAS GRANTED AN
IRREVOCABLE NON-EXCLUSIVE
LICENCE ALLOWING THE NATIONAL
LIBRARY OF CANADA TO
REPRODUCE, LOAN, DISTRIBUTE OR
SELL COPIES OF HIS/HER THESIS BY
ANY MEANS AND IN ANY FORM OR
FORMAT, MAKING THIS THESIS
AVAILABLE TO INTERESTED
PERSONS.**

**L'AUTEUR A ACCORDE UNE LICENCE
IRREVOCABLE ET NON EXCLUSIVE
PERMETTANT A LA BIBLIOTHEQUE
NATIONALE DU CANADA DE
REPRODUIRE, PRETER, DISTRIBUER
OU VENDRE DES COPIES DE SA
THESE DE QUELQUE MANIERE ET
SOUS QUELQUE FORME QUE CE SOIT
POUR METTRE DES EXEMPLAIRES DE
CETTE THESE A LA DISPOSITION DES
PERSONNE INTERESSEES.**

**THE AUTHOR RETAINS OWNERSHIP
OF THE COPYRIGHT IN HIS/HER
THESIS. NEITHER THE THESIS NOR
SUBSTANTIAL EXTRACTS FROM IT
MAY BE PRINTED OR OTHERWISE
REPRODUCED WITHOUT HIS/HER
PERMISSION.**

**L'AUTEUR CONSERVE LA PROPRIETE
DU DROIT D'AUTEUR QUI PROTEGE
SA THESE. NI LA THESE NI DES
EXTRAITS SUBSTANTIELS DE CELLE-
CI NE DOIVENT ETRE IMPRIMES OU
AUTREMENT REPRODUITS SANS SON
AUTORISATION.**

ISBN 0-315-98532-1

Canada

Name CHRISTOPHER DAVID VIGER

Dissertation Abstracts International is arranged by broad, general subject categories. Please select the one subject which most nearly describes the content of your dissertation. Enter the corresponding four-digit code in the spaces provided.

PHILOSOPHY

SUBJECT TERM

0422

SUBJECT CODE

U·M·I

Subject Categories

THE HUMANITIES AND SOCIAL SCIENCES

COMMUNICATIONS AND THE ARTS

Architecture 0729
Art History 0377
Cinema 0900
Dance 0378
Fine Arts 0357
Information Science 0723
Journalism 0391
Library Science 0399
Mass Communications 0708
Music 0413
Speech Communication 0459
Theater 0465

EDUCATION

General 0515
Administration 0514
Adult and Continuing 0516
Agricultural 0517
Art 0273
Bilingual and Multicultural 0282
Business 0688
Community College 0275
Curriculum and Instruction 0727
Early Childhood 0518
Elementary 0524
Finance 0277
Guidance and Counseling 0519
Health 0680
Higher 0745
History of 0520
Home Economics 0278
Industrial 0521
Language and Literature 0279
Mathematics 0280
Music 0522
Philosophy of 0998
Physical 0523

Psychology 0525
Reading 0535
Religious 0527
Sciences 0714
Secondary 0533
Social Sciences 0534
Sociology of 0340
Special 0529
Teacher Training 0530
Technology 0710
Tests and Measurements 0288
Vocational 0747

LANGUAGE, LITERATURE AND LINGUISTICS

Language 0679
 General 0289
 Ancient 0290
 Linguistics 0291
 Modern 0401
Literature 0294
 General 0295
 Classical 0297
 Comparative 0298
 Medieval 0316
 Modern 0591
 African 0305
 American 0352
 Asian 0355
 Canadian (English) 0593
 Canadian (French) 0311
 English 0312
 Germanic 0313
 Latin American 0314
 Middle Eastern 0313
 Romance 0314
 Slavic and East European 0314

PHILOSOPHY, RELIGION AND THEOLOGY

Philosophy 0422
Religion 0318
 General 0321
 Biblical Studies 0319
 Clergy 0320
 History of 0322
 Philosophy of 0469
Theology 0323

SOCIAL SCIENCES

American Studies 0323
Anthropology 0324
 Archaeology 0326
 Cultural 0327
 Physical 0310
Business Administration 0272
 General 0770
 Accounting 0454
 Banking 0338
 Marketing 0385
Canadian Studies 0501
Economics 0503
 General 0505
 Agricultural 0508
 Commerce-Business 0509
 Finance 0510
 History 0511
 Labor 0358
 Theory 0366
Folklore 0351
Geography 0578
Gerontology 0578
History 0578
 General 0578

Ancient 0579
Medieval 0581
Modern 0582
Black 0328
African 0331
Asia, Australia and Oceania 0332
Canadian 0334
European 0335
Latin American 0336
Middle Eastern 0333
United States 0337
History of Science 0585
Law 0398
Political Science 0615
 General 0616
 International Law and 0617
 Relations 0617
 Public Administration 0814
Recreation 0452
Social Work 0626
Sociology 0627
 General 0938
 Criminology and Penology 0631
 Demography 0628
 Ethnic and Racial Studies 0629
 Individual and Family 0630
 Studies 0700
 Industrial and Labor 0344
 Relations 0709
 Public and Social Welfare 0344
 Social Structure and 0709
 Development 0999
 Theory and Methods 0453
Transportation 0453
Urban and Regional Planning 0453
Women's Studies 0453

THE SCIENCES AND ENGINEERING

BIOLOGICAL SCIENCES

Agriculture 0473
 General 0285
 Agronomy 0475
 Animal Culture and 0476
 Nutrition 0359
 Animal Pathology 0478
 Food Science and 0479
 Technology 0480
 Forestry and Wildlife 0817
 Plant Culture 0777
 Plant Pathology 0746
 Plant Physiology 0306
 Range Management 0287
 Wood Technology 0308

Biology 0309
 General 0379
 Anatomy 0329
 Biostatistics 0353
 Botany 0369
 Cell 0793
 Ecology 0410
 Entomology 0307
 Genetics 0317
 Limnology 0416
 Microbiology 0433
 Molecular 0821
 Neuroscience 0778
 Oceanography 0472
 Physiology 0786
 Radiation 0760
 Veterinary Science 0760
 Zoology 0760

EARTH SCIENCES

Biogeochemistry 0425
Geochemistry 0996

Geodesy 0370
Geology 0372
Geophysics 0373
Hydrology 0388
Mineralogy 0411
Paleobotany 0345
Paleocology 0426
Paleontology 0418
Paleozoology 0985
Palynology 0427
Physical Geography 0368
Physical Oceanography 0415

HEALTH AND ENVIRONMENTAL SCIENCES

Environmental Sciences 0768
Health Sciences 0566
 General 0300
 Audiology 0992
 Chemotherapy 0567
 Dentistry 0350
 Education 0769
 Hospital Management 0758
 Human Development 0982
 Immunology 0564
 Medicine and Surgery 0347
 Mental Health 0569
 Nursing 0570
 Nutrition 0380
 Obstetrics and Gynecology 0354
 Occupational Health and 0381
 Therapy 0571
 Ophthalmology 0419
 Pathology 0572
 Pharmacology 0382
 Pharmacy 0573
 Physical Therapy 0574
 Public Health 0574
 Radiology 0575
 Recreation 0575

Speech Pathology 0460
Toxicology 0383
Home Economics 0386

PHYSICAL SCIENCES

Pure Sciences 0485
Chemistry 0749
 General 0486
 Agricultural 0487
 Analytical 0488
 Biochemistry 0738
 Inorganic 0490
 Nuclear 0491
 Organic 0494
 Pharmaceutical 0495
 Physical 0754
 Polymer 0405
Mathematics 0605
Physics 0986
 General 0606
 Acoustics 0608
 Astronomy and 0748
 Astrophysics 0607
 Atmospheric Science 0798
 Atomic 0759
 Electronics and Electricity 0609
 Elementary Particles and 0610
 High Energy 0752
 Fluid and Plasma 0756
 Molecular 0611
 Nuclear 0463
 Optics 0346
 Radiation 0984
 Solid State 0984
Statistics 0984

Applied Sciences 0346
Applied Mechanics 0984
Computer Science 0984

Engineering 0537
 General 0538
 Aerospace 0539
 Agricultural 0540
 Automotive 0541
 Biomedical 0542
 Chemical 0543
 Civil 0544
 Electronics and Electrical 0348
 Heat and Thermodynamics 0545
 Hydraulic 0546
 Industrial 0547
 Marine 0794
 Materials Science 0548
 Mechanical 0743
 Metallurgy 0551
 Mining 0552
 Nuclear 0549
 Packaging 0765
 Petroleum 0554
 Sanitary and Municipal 0790
 System Science 0428
Geotechnology 0796
Operations Research 0795
Plastics Technology 0994
Textile Technology 0994

PSYCHOLOGY

General 0621
Behavioral 0384
Clinical 0622
Developmental 0620
Experimental 0623
Industrial 0624
Personality 0625
Physiological 0989
Psychobiology 0349
Psychometrics 0632
Social 0451

**The undersigned recommend to the Faculty of Graduate Studies
and Research acceptance of the thesis**

**"A CLASSIFIER SYSTEM-BASED MODEL
OF SOME CHARACTERISTIC FEATURES OF CONSCIOUSNESS"**

**submitted by Christopher D Viger
in partial fulfilment of the requirements for
the degree of Master of Arts**

Thesis Supervisor

Chair, Department of Philosophy

CARLETON UNIVERSITY

September 7, 1994

ABSTRACT

In this essay I develop a mechanistic model of some characteristic features of human consciousness. The model is a development of the classifier system designed by John Holland. I analyze Holland's classifier system from a cognitive perspective and then add mechanisms to it to make the system more human-like in its cognitive activities. The first addition I make to the classifier system is to introduce subsystems into the architecture. Next I add pass through characters, which make a system able to perform abstraction and form representations. I also add predictor rules, my original contribution, making the system a fast learner and able to anticipate actions in its environment. Finally, I add second-order rules to the model, also my own contribution, which provide the system with something like self-awareness and unified cognitive states.

ACKNOWLEDGEMENTS

First I wish to thank my supervisor Dr J.A. Brook for his patience, faith, and good sense. Without his input, this work would be little better than a collection of vaguely related ideas. His assistance in structuring the work and developing its content was invaluable.

I would like to acknowledge the contributions of Dr F. Oppacher. It was during one of his presentations that I came up with the general idea for this thesis.

The input of Dr T. Gannon, Dr R. Stainton, and T. Kenyon, MA, in early versions of this work was of great importance. They were all sorely missed in its final stages.

As always, my parents completely supported me in this work and took an active interest in it, and as always their support is very much appreciated.

A special thanks is needed for my godparents Vi and Bob Harvey. Their "office" facilities and ever-willingness to have an ear bent were of more help than they could ever know.

Also, my parents-in-law have been nothing but encouraging and their positive attitude is greatly appreciated.

Finally, I wish to thank my wife Judith Gorman, MSc. To try to quantify her contributions to this work would undermine them. Without her none of this would have been possible.

TABLE OF CONTENTS

INTRODUCTION THE PHILOSOPHICAL PROBLEM	1
CHAPTER 1 ESTABLISHING A THEORETICAL FOUNDATION	7
1.1 Genetic Algorithms	7
1.1.1 What Is A Genetic Algorithm?	7
1.1.2 Coding and Evaluation: How a GA Structures the Problem	8
1.1.3 Reproduction, Crossover, and Mutation: The Genetic Operators	12
1.1.4 A Sample GA at Work	15
1.1.5 Schemata: A GA's Key to Success	17
1.1.6 Other Features of GAs	20
1.2 Classifier Systems	21
1.2.1 The Rule and Message System: Competing for Space	23
1.2.2 The Bucket Brigade: Paying Debts and Being Rewarded	25
1.2.3 The GA: Generating New Classifiers	27
CHAPTER 2 A COGNITIVE LOOK AT C1	30
2.1 Learning: Slowly but Surely	30
2.1.1 Coupling and Sequence Formation	30
2.1.2 Default Hierarchies	34
2.1.3 The Effect of Sharing the Rewards	37
2.1.4 Plasticity: The Substance of Learning	38
2.1.5 General Features of C1 Learning	40
2.2 Representations: How C1 "Sees" the World	41
2.2.1 What Constitutes a Representation	41
2.2.2 C1's Lack of Representations	44
2.2.3 An Instrumentalist Methodological Concern	47

2.3 Rewards	49
2.4: Concluding Remarks About C1	54
CHAPTER 3 C2: DECENTRALIZING CONTROL	55
3.1 Restricted Mating	55
3.2 Subsystems: Cobbling Together Basic Classifier Systems	58
3.3 A Closer Look at Subsystem Interaction	60
3.4 The Role of Schemata in C2	62
3.5 C2's Cognitive Features	63
CHAPTER 4 C3: ABSTRACT THINKER, QUICK LEARNER	66
4.1 C3's Information System	67
4.1.1 Pass Through Characters	67
4.1.2 Semantic Content: The Formal Approach	68
4.1.3 Abstraction: A First Look at Forming Conceptual Categories	74
4.1.4 C3's Representations; and Abstraction Again	78
4.1.5 Content in C3	81
4.2 Learning in C3	82
4.2.1 Predictor Rules	82
4.2.2 Using Predictor Rules for Single-Instance Learning	87
CHAPTER 5 C4: HERE WE ARE	90
5.1 A Mechanistic Look at Consciousness	91
5.1.1 Defining Second-Order Rules	91
5.1.2 The Role of Second-Order Rules in C4	93
5.1.3 Two Routes for Determining Action	96
5.1.4 C4-Consciousness	98

5.2 What is C4-Consciousness Like?	100
5.2.1 The Subject of C4-Consciousness	101
5.2.2 The Unity of Consciousness at a Time	102
5.2.3 Memory and Unity of Consciousness Across Time	105
5.3 A Walk in the Park with C4	106
5.3.1 Abstraction: A Final Look	106
5.3.2 Making Predictions	108
5.3.3 Heading for Home	109
5.4 Conclusion	110
BIBLIOGRAPHY	112

INTRODUCTION

THE PHILOSOPHICAL PROBLEM

I started the day by taking my dogs for a walk at an old gravel pit converted into a dog walking park. Since dogs can run free in this park it is quite a popular spot, so there were many other people walking their dogs also. As often happens, seven or eight dogs met up with each other and then began running and barking together. While I was watching this, I identified all of the dogs as dogs, i.e., as members of some category that I refer to as "dog". Some of the dogs were small, others large; they were different colours from each other; some had long hair, others short hair; yet, despite the considerable differences between the individuals, I classified them as members of the same group. Furthermore, I was aware that I was doing it; I was conscious of categorizing them all as dogs, and I was aware of myself as the one doing the categorizing. I was aware of seeing, hearing, smelling, and thinking about the dogs, and I was conscious that it was me, the same me, doing so in each case.

As I continued my walk through a wooded area, I saw some squirrels nearby. I predicted that my dogs would chase the squirrels when they saw them, which they did, but that they would not catch the squirrels, which they did not. While the dogs were running after the squirrels a branch fell from a tree and I instinctively moved out of the way so it would not hit me. It is interesting that if I had consciously thought about getting out of the way of the branch, I would not have moved in time to avoid being hit, yet of course I was conscious of the branch coming toward me. My instinctive reaction to the falling branch was much quicker than consciously thinking: "A branch is falling. It could hit me. I should move."

When we left the woods I saw a dog by itself. Knowing there was an animal hospital nearby, I thought to take it there until its owners came to get it since it was a cold morning. Unfortunately, I was unable to catch the dog, so eventually I gave up trying. I continued with my walk but I became preoccupied with the dog running loose by itself. I thought for some time about how to catch a dog in a similar situation and finally came up with the idea of carrying dog cookies with me whenever I walk my dogs so that

in the future I could use a dog cookie to entice the dog to come close enough to put a leash on it. As I returned to the parking lot I saw that the dog that had been loose was now happily walking on a leash beside someone, so I turned my thoughts to other things, namely my thesis. It's been a rather ordinary day so far.

During my rather ordinary walk in the park, I was engaged in some rather *extraordinary* cognitive activities, which upon reflection seem to be virtual miracles of nature that humans perform every day without giving them a second thought. In particular, I displayed many characteristic features of consciousness, and though perhaps the most distinctive feature of consciousness is its resistance to definition, there are certain characteristics that are distinctive of consciousness, without which a thing could not be said to be conscious. Firstly, conscious experiences have a certain content to them; we can simply be aware of something, as I was when I avoided the falling branch. We can also be self-aware, a more distinctly human experience, as I was when I was aware that *I* was seeing the dogs. Secondly, consciousness is unified. All of our mental states are experienced by the same "I" as a single "object". We have a unified subject of awareness and this subject persists over time. When I saw, heard, smelled, and thought about the dogs, it was the same "I" doing so in each case, the "I" that is now aware of writing a thesis. A third feature of consciousness is that a small number of things seem to dominate it at any one time. We can be aware of many things at any moment, but we can only consciously focus on a few of them at once. This was apparent when I was thinking about how to catch the dog. I was still aware of my surroundings since I avoided walking into puddles or trees, but my consciousness was dominated with thoughts about catching a loose dog; "my mind was somewhere else". Fourthly, conscious determination of an action is relatively slow compared with an instinctive reaction. I was only able to avoid the falling branch because I reacted instinctively. Left to my conscious deliberation I would have been hit. So at the very least we have that human consciousness consists of special content that includes the conscious subject (self-awareness), it has a unified subject at and across time together with a unified object (unity), it is relatively slow in determining actions, and it is dominated by thoughts about only a small number of things at any

one time (focused). In addition, consciousness seems to allow us to categorize objects, make predictions, learn how to act in a situation based only on one encounter with the conditions of that situation, merely by thinking our way through it, and plan and make decisions. The question is how. How is it that we are conscious? How do we do the things that consciousness seems to empower us to do?

When theorists attempt to explain consciousness, they either give up in frustration or do so in terms of consciousness. Of course, that is more than just a little question begging. This non-answer about the inner workings of our mind results from the way we experience our mental states. Intuitively, consciousness seems to be something like having a mind's "eye" such that we become conscious of whatever it casts its gaze upon. It is as if there is a little person inside of our heads, a homunculus, an "eye that sees" and a "thing that does". This is an extremely compelling image because that is exactly how consciousness seems to be. The image is compelling for another reason. Mental states have intentionality or "aboutness", i.e., they are about something. But things that are about something are usually about something *to* someone, a homunculus; "wherever a theory relies on a formulation bearing the logical marks of intentionality, there a little man is concealed" (Dennett 1978, p.12). The image is so compelling that it creeps its way into most theories of mind. For example, Roger Penrose seems committed to the idea. "Perhaps, when we refer to the *unconscious* mind we have a vague image of 'someone back there' who acts behind the scenes but who does not usually ... directly impinge upon what we perceive. Perhaps, the unconscious mind actually *has* an awareness of its own, but this awareness is normally kept quite separate from the part of the mind that we usually refer to as 'us'" (Penrose 1989, p.524). Though Penrose does not explicitly refer to a "someone in there" giving rise to consciousness, this is certainly implied by the connection he draws between "someone back there" and awareness. Even Paul Churchland, who is about as active a foe of homuncularism as one can be, cannot escape the homuncular picture when describing our introspective self-awareness: "... enhancement in our *introspective vision* could approximate a revelation. Glucose consumption in the forebrain, dopamine levels in the thalamus, the coding vectors in specific neural path-

ways, resonances in the nth layer of the peristriatal cortex, and countless other neurophysiological and neurofunctional niceties could be moved into the *objective focus of our introspective discrimination and conceptual recognition*, just as Gmin7 chords and A+9 chords are moved into the objective focus of a trained musician's auditory discrimination and conceptual recognition" (Churchland 1988, p.180, emphasis mine).

Despite its intuitive appeal, the homuncular image of consciousness has no explanatory value in a theory of mind. Homunculi creep into theories where explanations fail, they do the things that the theory cannot account for. A materialist theory of the mind must explain how the brain's activities can lead to the rich mental life we experience as humans. It is quite acceptable for a theorist to defer answering a particular problem, provided she flags it as a problem; however, the introduction of a homunculus, implicitly or explicitly, is tantamount to positing a miracle. The problem is that all of the interesting work is done by the homunculus. We are conscious of something when the homunculus is aware of it, and the homunculus determines our actions. But a theory of mind should account for how we are conscious and how consciousness allows us to do the things we do, so the same questions persist, they are merely transferred onto the homunculus. How is the *homunculus* aware of anything and how does *it* determine actions? Clearly a miniature homunculus in the "brain" of our homunculus will lead to an unacceptable infinite regress of homunculi. Many theorists avoid an infinite regress by simply including an undischarged homunculus in their model. This is, of course, a weak spot in their models. A complete theory must discharge a homunculus at some point, yet if any of the homunculi can be discharged then it would seem that they all can be. Thus, no complete theory of mind should include a homunculus. "The account of intelligence required ... must not explain intelligence in terms of intelligence, for instance by assigning responsibility for the existence of intelligence in creatures to the munificence of an intelligent Creator, or by putting clever homunculi at the control panels of the nervous system" (Dennett 1978, pp.72-3).

Identifying the problem with a homuncular picture of the mind is the easy part. The difficult part is exorcising it from our models. It is essential to a materialist theory

of the mind that it be purely mechanistic, and that the mechanisms not be intelligently guided. One possibility in this direction is to decentralize the model. A homunculus is a focal point of consciousness, so having a decentralized model might help to avoid the homuncular image. This approach is at least plausible since the brain seems to be massively parallel with no control centre. The idea is that at least some interesting mental work must occur in the brain's subsystems, without that work being "intelligently" delegated to the subsystems. As specialists, the subsystems must "know" enough to do their jobs without being told when or how to do them. Perhaps the strongest advocate of this approach in the philosophical literature is Dennett, who suggests that the mind might operate on the basis of some sort of pandemonium in which subsystems usurp control. (Dennett 1991). The purpose of this essay is to present a non-homuncular mechanistic model of some of the features of consciousness discussed above. In other words, the model will keep the functions of a homunculus without using one to do so. In order to avoid letting a homunculus slip back in unawares, I will take a bottom up approach, beginning with a computational system developed by John Holland called a classifier system. First I will give a theoretical presentation of a genetic algorithm, on which a classifier system depends, and then the classifier system itself. I will then analyze a classifier system from a cognitive point of view. Since the objective is to produce a mechanistic model of some characteristic features of consciousness, I will develop new systems by adding mechanisms to a classifier system that will augment its cognitive abilities. This process will be a kind of reverse engineering in which knowledge of certain features of the required model can guide its development. The development of a classifier system will occur in four stages as a metaphorical evolution of a creature, C1, with a classifier system brain. The first stage is C1. Then subsystems will be introduced, so that the resulting system, C2, can specialize its cognitive resources. C2 needs a way to coordinate the actions of its subsystems, and the way it does so is an extremely important mechanism for modelling features characteristic of consciousness. The third system, C3, is developed from C2 by adding two mechanisms: pass through characters and predictor rules. It has the important features that it can form representations, perform abstraction,

and learn quickly. The final system, C4, adds second-order rules to the cognitive architecture of C3. These allow it to form representations of its representations, which is how C4 possesses certain features that are characteristic of consciousness.

It is important to note that my model will not provide an account of volition, the conscious determining of an action. That is to say, I will not consider the nature of the will. Also, the objective is to produce a model with mechanisms that capture distinctive features of consciousness. I am not claiming that the presence of these mechanisms is either necessary or sufficient for consciousness, merely that they provide a system with features characteristic of it. In this way they might serve as a mechanistic underpinning for consciousness.

CHAPTER 1

ESTABLISHING A THEORETICAL FOUNDATION¹

1.1 Genetic Algorithms

1.1.1 What Is A Genetic Algorithm?

In the late 1960's and early 1970's John Holland noticed that such diverse fields as genetics, economics, control theory, game theory, psychology, and artificial intelligence share a common problem: How is optimization possible in a very complex and uncertain environment? Natural systems manage to exploit information as they acquire it so that their behaviour is modified by feedback from their environments, i.e., they adapt. Moreover, they quickly disseminate "good tricks" throughout the next generation via natural selection. Holland used the processes behind natural selection and evolution as a model for an algorithm that could be implemented in artificial systems facing similar optimization problems to natural systems. The result was the genetic algorithm, named for the field that inspired it.

A fundamental feature of natural evolution is that it takes place on chromosomes and not on the living beings they encode. Natural selection may then be conceived as the tendency for chromosomes that encode successful creatures to be reproduced more often than those encoding unsuccessful creatures. This is one important idea that Holland incorporated into genetic algorithms. Another important aspect of natural evolution that guided the development of genetic algorithms is that evolution occurs during reproduction by processes of recombination and mutation. While mutations do cause the chromosomes of children to differ from those of their biological parents, it is through the process of recombination, in which genetic material of the biological parents is exchanged, that most of the significant diversity between the chromosomes of biological parents and their children results. Thus recombination or crossover is the main driving mechanism behind genetic algorithms. A final interesting feature of natural evolution that must have been

¹ My presentation of genetic algorithms and classifier systems is based on Goldberg 1989, and incorporates ideas from Holland 1975, Holland 1992, Holland et al 1987, and Davis 1991. None of the mechanisms in this chapter is my original work, but I did generate the examples.

very appealing to the computer scientist in Holland is that it has no memory other than what is implicitly contained in the existing population. What evolution "knows" about producing individuals that can successfully cope with their environments is stored exclusively in the chromosomes of those individuals that have survived. Ostensibly intelligent decisions are made by simple mechanisms. Clearly this feature is of central concern to our investigation and it will be discussed in detail below.²

1.1.2 Coding and Evaluation: How a GA Structures the Problem

Now that we have seen the motivation for genetic algorithms (GA's) from natural evolution, we are in a position to consider the general structure of GA's in more detail. All genetic algorithms have certain features in common which are independent of the particular optimization problems for which they are designed. It will be sufficient for our purposes to consider only the most basic structures that comprise a GA. Of course, not all of the features of a GA can be independent of the problem for which it is to be used. In fact, there are exactly two features of a simple genetic algorithm that are problem specific, the coding the GA uses and its fitness or evaluation measure, which is just a function over the encoded possible solutions in a simple GA.

How to determine appropriate codings for a given problem can be a very difficult matter, but for our purposes we will assume that any possible solution to the problem can be represented by a binary bit string of constant length.³ These strings play the role of chromosomes in nature and are sometimes referred to as such. A very simple example of such a binary coding would arise in a problem involving a system of pipes with ten valves to control water flow through the pipes. Suppose pressure in the pipes depends in some nonlinear way on which valves are open and pressure must be maximized to ensure the greatest flow. A natural coding for this problem would be binary strings of length ten.

² The features of natural evolution that motivated Holland to create genetic algorithms are listed in Davis (1991) and my discussion is based on that list.

³ This assumption is an oversimplification but since the specific coding will not be relevant to our investigation I will avoid the subtleties involved in choosing a coding. Also there are advantages to using a binary coding whenever possible because it increases the number of schemata present in any given generation. The importance of this will become clear when we discuss schemata below.

Each position in the string would be assigned a correspondence with one of the valves, a 1 indicating that the valve is open and a 0 meaning it is closed. The problem thus reduces to finding the string that will yield the best result; however, this also requires a notion of "best".

To determine which of the strings is the best solution to the given problem, some kind of evaluation or determination of the strings' fitnesses must be performed, which must rely on the specific problem being solved. In the simplest cases the evaluation is performed by a function, whose domain is the encoded binary strings that represent possible solutions to the problem and whose range is the non-negative real numbers, such that the fittest or best solutions are those that return the highest number after evaluation. In our water pipe example, there could be a function -- determined theoretically or experimentally -- that gives the water pressure in the pipes, for a certain arrangement of the valves being open or closed, when the string corresponding to that arrangement is evaluated. For example, we might have the evaluator function $E(x) = (x-7)^2$ for $x \geq 1$ and $\ln(x+1)$ for $x < 1$, where x is the number obtained by treating the bit strings as binary numbers that are converted to decimal numbers and divided by a scaling factor of 100. Of course, this function is merely a pedagogical tool and would not be used for any real evaluator function. Determining the evaluator function for a given problem can be a very difficult matter requiring experimental and theoretical knowledge. A good indication of what can be involved is given in a paper *Genetic Algorithms in Parametric Design of Aircraft* by Bramlette and Bouchard, included in Davis 1991, pp.109-123. Their problem requires fifty parameters, and they must include knowledge of the geometry of the design, aerodynamic properties such as lift and drag, fuel consumption, and weight by component, for example. The point is that being able to determine a function to evaluate how good a particular potential solution is does not entail knowing the best solution. Classifier systems avoid the problem of having to determine an evaluator function because nature does this for them in the form of rewards, as we will see.

Returning to our example, the string 1001000010, which corresponds to the situation in which the first, fourth, and ninth valves are open and the rest are closed, gets

evaluated to be 1.4884. This evaluation is referred to as the string's fitness, which in our problem corresponds to the water pressure in the pipes. Taken as a binary number, the string 1001000010 represents the decimal number 578, which scales to 5.78 in our example. The fitness for the string is obtained by letting $x = 5.78$ in the evaluator function; i.e., $E(5.78) = (5.78-7)^2 = 1.4884$. Similarly, the arrangement in which the second, third, fifth and tenth valves are open and the rest are closed is encoded by the string 0110100001, which has a fitness value $E(4.17) = (4.17-7)^2 = 8.0089$. The arrangement in which only the sixth valve is open is represented by the string 0000010000 which has a fitness value $E(0.16) = \ln(1+0.16) \approx 0.1484$. In this case we had to use the natural logarithm part of the definition of $E(x)$ because the value of x was less than 1. As we would expect, the pressure when all the valves are closed is 0; the string 0000000000 is 0 in decimal notation and $E(0) = \ln(0+1) = 0$. There are a total of $2^{10} = 1024$ possible arrangements of the valves being open or closed, each having a unique ten bit binary string representation whose fitness can be determined using the function $E(x)$, as above. The desired arrangement in this problem is the one corresponding to the string with the highest fitness, i.e., the string that yields the highest number after being evaluated by the function, $E(x)$.

It might seem that the existence or discovery of some function to evaluate the strings entails the solution to the problem being considered. This is not the case, however, for the problem is to find an optimal or near optimal solution, which for our purposes is simply a string with the maximum or greatest value in the range of the evaluator function. In other words, an optimal solution is any string whose fitness is not less than the fitness of any other string representing a possible solution; however, being able to determine a value for any one possible solution in no way entails knowledge of any optimal solutions. Solution spaces⁴ in all but the most trivial problems are extremely complex, so that even if we have an explicit function that we can evaluate at any point in the domain of possible solutions, we cannot, from that alone, determine the topology of the solution space.

⁴ By solution space, I mean the Cartesian cross-product of the range and domain of the evaluator function. In our case, this will be $\{0,1\}^l \times \mathbb{R}$ when the strings are of constant length l .

Moreover, the maximum value of the function, which corresponds to an optimal solution, is not something we know. It is not the case that we have a value we know to be maximal and we are looking for strings whose fitness equals that value or is close to it. When we evaluate a particular string, our only sense of how "good" it is, is a relative one, obtained by comparison of its fitness value to the fitnesses of the other strings we have evaluated. This is a part of the reason why optimization is so difficult.⁵ For our purposes we are assured that there is at least one optimal solution, though there may be many, because the domain is finite -- usually enormous, but finite nonetheless -- and each string has a definite value under the evaluator function. The difficulty is finding one.

In our simple valve example the evaluator function, $E(x)$, allows us to calculate the pressure for any given configuration of valve openings, but this by itself does not help us decide which is the best configuration. Given any of the over one thousand strings in our problem, we can find the pressure value for its corresponding arrangement. This alone cannot tell us which string has the highest value. For that we must still search the solution space for an optimal solution; the evaluation function merely allows us to test our solutions theoretically rather than practically, which is in itself a large advantage; however, we still require some efficient means of searching. Clearly in our simple problem an exhaustive search of the solution space is possible because there are only 1024 possible solutions. However, if the problem involved forty or fifty valves there would be in excess of one trillion possibilities, and many problems have solution spaces larger than that. Any optimization algorithm requires some means of distinguishing good solutions from bad ones, which requires problem specific information; but, using the operations of recombination and mutation found in natural evolution, simple genetic algorithms require no other problem specific information. The rest of the process is very much a blind

⁵ This is the sense in which a GA is a blind search, meaning that it "use[s] only the structure of the space of alternatives in selecting the next alternative to consider; any "legal" possibility is as worthy of consideration as another" (Stillings et al 1991, p.171). It is worth noting a distinction between a blind search and a random search. Blind searches follow some algorithm to obtain better than random search results, so a blind search is guided. "The blind search methods do bias the exploration one way or another" (Stillings et al 1991, p.184). What is blind about the search is that the only available information about the structure of the solution space is what has been discovered by searching.

search. We now turn to those features that uniquely characterize genetic algorithms.

1.1.3 Reproduction, Crossover, and Mutation: The Genetic Operators

Once possible solutions can be encoded into binary strings, say of length l , and there is a function by which these strings can be evaluated, the genetic algorithm begins by producing a population of strings, say n in number, where n is even. The first n strings are produced totally at random and stored as the initial population or first generation. Each chromosome or string in the first generation is evaluated and given the chance to reproduce in proportion to the ratio of its fitness to the fitness of the entire population, which is the sum of their individual fitnesses. Returning to our example, let us suppose that the four strings we evaluated earlier comprise our randomly generated initial population. Thus we have string₁ = 1001000010 with fitness 1.4884, string₂ = 0110100001 has fitness 8.0089, string₃ = 0000010000 has fitness 0.1484, and string₄ = 0000000000 has fitness 0. The fitness of the entire population is the sum of the strings' individual fitnesses which equals $1.4884 + 8.0089 + 0.1484 + 0 = 9.6457$. The probability of each string reproducing is the ratio of its fitness to the fitness of the population, so string₁ has a probability of $1.4884/9.6457 = 0.1543$ of reproducing. The other probabilities are calculated in the same manner and the results are summarized below.

<u>Initial Population</u>	<u>Fitness</u>	<u>Probability of Reproducing</u>
String ₁ : 1001000010	1.4884	0.1543
String ₂ : 0110100001	8.0089	0.8303
String ₃ : 0000010000	0.1484	0.0154
String ₄ : 0000000000	0.0000	0.0000

This type of reproduction operator is known as the roulette wheel because it amounts to spinning a biased roulette wheel to determine which strings from the first generation will reproduce.

A new population of n strings is created using the roulette wheel technique. Notice that strings with very high relative fitness may occur in the new population several times, whereas strings with very low fitness may not occur at all. This new population is not the next generation, after all it consists entirely of individuals from the first generation.

Rather, it represents how many times the individuals in the first generation will be allowed to reproduce. It is perhaps more appropriate to label it the parent population, since it contains only those members of the initial population chosen to reproduce, one copy of each string for each time it will mate. The very fit individuals will produce many offspring and the unfit individuals will produce few or no offspring. This is the genetic algorithm implementation of natural selection. Only the strong, i.e. fit, survive to reproduce and the weak become extinct. In our example, string₄ has no chance of reproducing, consistent with our intuition that closing all of the valves will not lead to the highest pressure and is, in fact, so far from a useful solution that we chose to totally disregard it. String₂ has a very high chance of reproducing, whereas string₁ has a slight chance and string₃ has almost no chance of reproducing. For the sake of our example, however, let us suppose that random processes result in a parent population containing two copies of string₂, one copy of string₁, and one copy of string₃.

The next generation is produced from the parent population by recombination, or crossover as it is more commonly referred to in the GA literature, and mutation. The strings in the parent population are randomly paired -- this is the reason for choosing n even. For each pair, a number between 1 and $\ell-1$, say k , is chosen at random as a crossover site. The parent strings are then altered by exchanging their bits after the k^{th} position, producing two children. Referring again to our valve example, we might have one copy of string₂ in the parent population paired to mate with string₁, and the other paired with string₃. Suppose that the randomly chosen crossover site for the first pairing is 7. In this case the first seven bits of each parent are left unaltered and the last three are exchanged to produce offspring as shown below.

(Parent)string₁: 1001000|010 Child₁: 1001000|001

(Parent)string₂: 0110100|001 Child₂: 0110100|010

If the randomly chosen crossover site for the second pairing is 2, only the first two bits of each parent are left unaltered and the remaining eight are exchanged, as shown.

(Parent)string₂: 01|10100001⁶

Child₃: 01|00010000

(Parent)string₃: 00|00010000

Child₄: 00|10100001

Notice that if the two parents have the same value for a particular bit position then both children will also have that same value in that bit position. For example, in the second pairing both parent strings begin with 0's and the children strings produced by crossover also both begin with 0's. This can cause a problem for the algorithm if the entire population comes to acquire the same value for some position, particularly if many good solutions have the opposite value for that position. This difficulty is avoided by mutation. Once crossover is complete, every bit in the new strings is tested to see if it mutates. This is achieved by choosing a mutation rate, usually something quite small such as 0.005, and then for each bit choosing a random number between 1 and 1000. Those bits for which the number is 5 or less mutate, i.e., they change from a 1 to a 0 or vice versa,⁷ and the rest remain the same. In our example, it might occur that only bit 7 of child₃ mutates. In that case child₃ is transformed from the string 0100010000 into the new string 0100011000, the seventh bit having changed from a 0 to a 1. After crossover and mutation are complete⁸ the children are "born" by being placed into a new population that will become the next generation -- literally they are stored in an array that represents the new population. Once n children have been created, reproduction stops and the new population replaces the previous generation from which it was produced -- the array containing the old population is deleted by making the new population the current generation. In GA's, populations make a complete changeover each generation. Once a new generation is produced the old one "dies off". The entire process is then repeated

⁶ This is, of course, the second copy of string₂ in the parent population, and might be more appropriately labelled parent₂; however, I have kept the name string₂ to emphasize its origin.

⁷ In some applications the value of mutation is predetermined so that some "mutated" bits actually remain the same. On average this would occur about half the time, so to achieve the same result as the method I described the mutation rate for this procedure would have to be doubled.

⁸ There are many other operators besides just mutation and crossover that can be used in a GA, but these are all that are required in a basic GA. The most noteworthy omission in this account is the inversion operator, which is described in Holland 1975, pp.106-9 and Davis 1991, p.21.

until a predetermined number of generations has been produced.

1.1.4 A Sample GA at Work

Let us see how the entire process works by producing five generations of strings in our valve problem. We will begin with same initial population and second generation produced above.

<u>Strings</u>	<u>Fitness</u>
First Generation	
1001000010	1.4884
0110100001	8.0089
0000010000	0.1484
0000000000	0.0000
	9.6457 total fitness
Second Generation	
1001000001	1.5129
0110100010	7.9524
0100011000	17.6400
0010100001	29.0521
	56.1574 total fitness
Third Generation	
0110101000	7.6176
0100010010	18.1476
0100011001	17.5561
0010100000	29.1600
	72.4813 total fitness
Fourth Generation	
0100010000	18.3184
0010100010	28.9444
0100100000	16.9744
0010011001	29.9209
	94.1581 total fitness
Fifth Generation	
0010100000	29.1600
0100100110	16.4836
0010111001	26.5225
0010000011	32.3761
	104.5422 total fitness

A few observations about this process are in order. We saw in detail how the second generation was produced from the first. Briefly, the remaining generations were produced as follows. From the second generation, one copy of the second string, two copies of the third string, and one copy of the fourth string were randomly selected to be in the parent population.⁹ Notice that the third string reproduced more often, even though the fourth string had a higher probability of reproducing, being the fitter of the two. The

⁹ My "random" generator consisted of dice throws with discretionary veto of results that would compromise the example.

second and third strings mated with a crossover site at 6, the third and fourth had crossover site 7, and there were no mutations. From the third generation, the second and fourth strings mated with crossover site 8, the third and fourth strings mated with crossover site 3, and again there were no mutations. Mating in the fourth generation occurred between the second and third strings with crossover site 4, and between the second and fourth strings with crossover site 5. The second offspring mutated in bit 8, and the fourth offspring mutated in bit 10. It is interesting to see the effects of mutation. In the initial population the seventh and eighth bits were 0 in every string. This would necessarily remain the case if crossover was the only operation, but because of mutation, diversity is introduced into these bit positions in the final generation. Such diversity is essential since without it the GA would not have access to some of the possible solutions, and depending on the randomly generated initial population, this could exclude many of the best solutions. What this example clearly demonstrates is how, even with a very unfit initial population, the population fitness significantly increases from one generation to the next; also, the maximum fitness of the strings in a generation increases.

More generally, an interesting feature of GA's is that it is possible for a string with very high fitness to survive from one generation to the next, the GA equivalent of old age. This could occur, for example, if two copies of the same string in the parent population are paired for crossover, as might have occurred in the case of string₂ from the initial population in our example. If no mutation occurs in such a case, duplicate copies of one string will be produced in the successive generation, GA identical twins. However, this possibility is greatly reduced when n and ℓ are large. Another way in which a string can survive into the next generation is if it paired with a string that is identical to it after the crossover site. In this case both strings will survive -- of course, the first situation is a special case of this more general situation. It is also possible that a string will reappear in a later generation just from the normal operations of the GA -- GA reincarnation -- as happened in our example, when the fourth string in the third generation reappeared as the first string in the fifth generation.

It is clear from the discussion above and from our example that the main operation

functioning in a genetic algorithm is crossover, but that is only because it was made to be so. The small effect of mutation is a result of setting a very low mutation rate. A much higher rate could have a significant impact on the strings produced for new generations. This would be undesirable, however, because it would greatly reduce the efficiency of the GA. What makes crossover a powerful optimization technique is that it combines partial solutions to the problem, thereby producing even better partial solutions, whereas mutation is tantamount to a random search. To see more clearly what crossover entails we must consider the notion of a schema.¹⁰

1.1.5 Schemata: A GA's Key to Success

The notion of a schema requires introducing a new character into our alphabet: *, for "don't care". The character * does not actually occur in any string in the population, that is, the coding is not being altered by this introduction, but it will serve to mean that either 0 or 1 is acceptable in the position it occupies. Thus the notation 1*0 actually means the two strings 110 and 100. Using this notation we can define a schema as a subset of strings from the domain of the evaluator function, identified by specifying definite values in certain string positions, while letting all other positions be either 0 or 1. For example the schema 1*...* would consist of all the strings beginning with 1, exactly half of the evaluator's domain; the schema *...* would consist of the entire domain of the solution space; 1...1* would consist of the two strings 1...11 and 1...10; and, any schema with all of its positions specified would consist of a single string.

The offspring of strings belonging to some schemata are very likely to belong to the same schema as their parents, whereas other offspring are very unlikely to be in the same schema as their parents. In particular, the greater the distance between the first fixed position and the last fixed position of any given schema, the more likely it is that crossover will result in children that are not members of the same schema as their parents.

¹⁰ "Schema" is a technical term from GA literature coined by Holland (Holland 1975). This term should not be confused with the notion of schema used in psychology. Holland et al use the term "substring schema" to avoid confusion (Holland et al 1986, p.119) but I use the term so often that in the interests of readability I will follow the tradition in the GA literature.

For example, there is only a $1/\ell-1$ chance that the child of a parent in the schema $1*...*0*$ will inherit the characteristics necessary to make it a member of that schema directly from its member parent. Of course, if both parents belong to the schema then, unless mutation occurs at one of the fixed bits, both children will also belong to that schema. Conversely, schemata that are very short are extremely stable and crossover almost always results in children that belong to the same schema as their parents. For example, there is only a $1/\ell-1$ chance that one child of a string in the schema $*...*00*...*$ would not be a member of the schema because the essential pattern was broken by crossover. These short schemata are very stable under crossover. Now if it is also the case that a schema with short defining length occurs in a string with very high fitness, it is extremely likely that the new generation will contain strings that belong to the same schema. These highly fit schemata with short defining length -- i.e. with a very short distance between the first fixed bit and the last fixed bit -- are the key to the success of genetic algorithms; they are given the name "building blocks".

Building blocks function to improve the overall fitness of the population in successive generations because they occur in strings of high fitness. These strings are the ones most likely to reproduce and their offspring are extremely likely to possess their building blocks. If it is accidental that a particular schema happens to occur in a string of high fitness, i.e., the schema itself does not lead to the high evaluation¹¹, then new strings containing that schema will eventually become relatively poor solutions and the schema will die out. However, if a schema is a partial cause of the high fitness of the strings in which it occurs, it will occur more often in successive generations because its strings will have high evaluations, leading to greater frequency of reproductions; i.e., strings containing that building block will be selected to the parent population increasingly

¹¹ A schema can be regarded as a cause of the fitness value of the string in which it is contained because the fixed bits that define the schema make a certain contribution to the evaluation, under the evaluator function. In fact, a schema is an encoding of some actual physical cause. It must be stressed, however, that the contribution of a schema or any fixed bit cannot be determined exclusively from that schema or bit because of the nonlinearity of the evaluator function. In our valve example, 0001000000 has a fitness of 0.4947 and 0001110000 has a fitness of 34.5744, yet both belong to the schema 0001*0000, which contains only four strings.

more often. The actual increase can be precisely formulated and this mathematical result is known as the Fundamental Theorem of Genetic Algorithms or the Schema Theorem.¹² The increase is geometrical in the ratio of the average fitness of all strings containing the building block to the average fitness of all strings in the population; but, the increase is modified by the effects of crossover and mutation, which diminish the increase by a derivable amount depending on the length of the schema and the number of fixed positions it contains.

Even in our simple example, we can see the GA exploiting building blocks. As early as the second generation, the GA has "identified" the schema 001***** and by the fifth generation three of the four strings are in this schema. This schema constitutes a building block because it has a very short defining length, only a crossover site of 1 or 2 could break it up during recombination, and the strings belonging to it are very fit -- the fitnesses of strings in this schema range from 19.8025 to 32.7184. It is interesting to note that the fourth bit converged to 0 in the third generation and no mutation occurred to introduce diversity. This is interesting because the optimal solution is the string 0001100100, which has a fitness of 36, and without a mutation the GA would be unable to find that solution. However, given that the fourth bit had converged to 0, the best possible solution is 0010000000 whose fitness is 32.7184, and very quickly the GA produced the string 0010000011 whose fitness is 32.3761. The GA performed just about as well as it possibly could "under the circumstances"¹³.

The schema 0001***** is a very interesting one because it straddles the discontinuity in the evaluator function. The schema contains sixty-four strings, thirty-six of which have fitnesses no larger than 0.4947 and twenty-eight of which have fitnesses no smaller than 32.8329. The optimal solution 0001100100 is contained in this schema, yet

¹² The formula and its derivation can be found in Goldberg 1989, pp.28-33, but it is beyond the scope of this paper to present them here.

¹³ Its performance was affected by the fact that the initial population was chosen to be far worse than expected for a randomly generated population -- the string 0000000000 literally contributed nothing to the search -- and the run was very short with an extremely small population.

the schema is not a building block because over half of its strings are among the most unfit in the population. Using the discontinuous evaluator function, $E(x)$, makes it clear that the contribution of a particular schema or bit cannot be determined in isolation. For example, what is the contribution of the fourth bit? The only possible answer, as we have just seen, is that it depends on what else is in the string. The fourth bit does contribute to high evaluations, it is contained in the best twenty-eight strings in the population, but it cannot cause high fitnesses on its own. In fact, in our example, the best strings are the ones that, when taken as a binary number, convert to a decimal number close to, but not less than, 100. Thus the fourth bit, which contributes 64 to the decimal number, must be 1 for high fitnesses, but the fifth bit must also be 1 as must one of the sixth, seventh, or eighth bits, and the first three bits must all be 0's for the very highest fitnesses. What this amounts to is that the schema 0001***** is of no use to the GA in its search -- that is to say, the emergence of this schema in the population will not result in an overall increase in the population's fitness; but, the schemas 000111****, 00011*1***, and 00011**1**, all contained in 0001*****, will increase total fitness if they appear in the population.

1.1.6 Other Features of GAs

Intuitively, the Fundamental Theorem says that each new generation builds on the useful information gained by the previous generation, combining small pieces of strings that lead to high fitness to produce better and better solutions to the problem -- i.e. strings that have higher and higher fitnesses approaching that of the optimal solution. The genetic algorithm requires no memory to store and exploit the information gained by a particular generation, other than the strings themselves, because the best building blocks are reproduced most often, and always in high valued strings. In this ingenious manner the entire accumulated "knowledge" of the genetic algorithm is stored in the population with nothing but a stochastic mechanism to implement this knowledge. There is no central structure in the GA that has any access to what it implicitly "knows", yet increasingly "intelligent" decisions are made in each successive generation.

Genetic algorithms qua optimizers are extremely robust, performing efficiently in

a wide variety of problem settings. One of the main reasons for this is that in the production of each new generation, GA's process many more schemata than they do strings. This is a result of the fact that each string of length ℓ is a member of 2^ℓ different schemata. As each new generation of n strings is produced, on the order of n^ℓ schemata are processed.¹⁴ Holland named this effect "implicit parallelism". Explicitly, a GA evaluates n strings from the total domain of all possible solutions to the problem, i.e., it samples strings n at a time. Contained in any sample string population of n strings there are anywhere from 2^ℓ to $n2^\ell$ distinct schemata, which are implicitly being evaluated along with the strings. Some of these schemata will increase in number in the next generation and some will decrease, independently of what happens to the other schemata, as given by the Fundamental Theorem of GAs. Implicit parallelism is the effect that despite there being only n strings, on the order of n^ℓ schemata increase in frequency in the next generation, and hence are sampled more often. The better a region of the solution space seems to be, based on existing samples, the more samples a GA takes in that region. The power behind a GA is the large number of schemata it can sample by evaluating a relatively small number of strings. Implicit parallelism occurs with no special bookkeeping or memory -- a GA does not need to keep an explicit account of the schemata it is sampling; rather, implicit parallelism arises from a blind search by sticking with what works. When a genetic algorithm happens to find a string with relatively high fitness, it has no idea why that string is good. But by testing, i.e. evaluating, schemata in that string in combination with parts of other strings, the GA is able to separate out the truly efficacious schemata in the string from patterns that are redundant, the latter being superseded by more successful schemata. Eventually the GA combines these high fitness substrings into near optimal solutions.

1.2 Classifier Systems¹⁵

Genetic algorithms were designed to operate in the same way as natural selection.

¹⁴ For a derivation of this estimate see Goldberg 1989, pp.40-1.

¹⁵ This presentation follows closely the discussion in Goldberg 1989, pp.221-230.

Both rely on the mechanistic processes of recombination and mutation to create a new population from an existing population, with a bias towards letting the most successful members of the population reproduce most often. This bias leads to a general improvement in the fitness of individuals in future generations. However, "fitness" is understood very differently in the two contexts. In the case of a GA, a problem specific evaluator function determines the fitness of the strings in the population. Only the relatively fit strings live to reproduce. Nature's evaluator function is survival itself; only the creatures that live to reproduce are fit. Nature's "problem" is to produce creatures that can survive, so the only evaluation of its success is the emergence of creatures that actually do survive.

Of course, nature is not really trying to solve any problem at all. A series of mechanistic processes occurs to produce various types of creatures, and the ones that live to reproduce are the fit ones. "Fitness" is defined after the fact in nature because there is no intelligent intervention at any stage of the process. Unlike the situation in a genetic algorithm, nature cannot predetermine what will constitute a survival advantage and then endow only those creatures having the requisite features for survival with the ability to reproduce. Whether or not wings and webbed feet, or large brains and opposable thumbs confer sufficient survival advantage to warrant incorporation of these features into future generations of creatures, is determined exclusively by whether or not the creatures having those features actually do reproduce. GA's operate on the principle: Is this string fit? -- as measured by the evaluator function. If so, let it reproduce. Natural selection works on the principle: If this creature -- the realization of a string of genetic code -- lives to reproduce, then it is fit. In order to produce a purely mechanistic model of the mind that is not question-begging, we will have to follow the principle that guides natural selection. Thus, we must not include any explicit evaluator functions in our model. What is required, then, is a system very much like a genetic algorithm, but without an evaluator function. Classifier systems are exactly that sort of system.

1.2.1 The Rule and Message System: Competing for Space

A classifier system has three main components. It consists of a rule and message system, a system for apportioning credit, and a genetic algorithm (Goldberg 1989, p.221). A message in a classifier system is just a binary string of constant length, exactly like those we have been considering above. A special class of rules with a very simple structure are called classifiers¹⁶ and these are the rules classifier systems use. They are strings that are separated into two parts of equal length, a condition part and a message part. The message part is itself a message, i.e., it is an element of the set $\{0,1\}^l$, which consists of all binary strings of length l . The condition portion of a classifier is a member of the set $\{0,1,\#\}^l$. The # character acts as a wild card; a classifier's condition is satisfied by a message when all of the following conditions are satisfied: if in every position in the condition part of the classifier that contains a 0, there is a 0 in that position of the message; if in every position in the condition that contains a 1, there is 1 in that position of the message; and if in every position in the condition that contains a #, there is a 0 or a 1 in that position of the message. For example, the message 10011 satisfies the condition of the classifier #0##1:00010, but it does not satisfy the condition of the classifier 0#01#:11100. In the first case every specified bit corresponds exactly to the bit in the same position of the message, whereas in the second example the message begins with a 1 and the condition begins with a 0. A classifier system uses classifiers as its only algorithmic device, and since classifiers are just a certain kind of rule¹⁷, classifier systems are a special type of production system, which just are exclusively rule-based systems. "A classifier system is a machine learning system that learns syntactically simple string rules (called *classifiers*) to guide its performance in an arbitrary environment" (Goldberg 1989, p.221).

A classifier system comes equipped with detectors that can obtain information

¹⁶ "The rules are called classifiers because they can be used to classify messages into general sets" (Holland et al 1987, p.103).

¹⁷ Since classifiers are the only kind of rule I am considering, I will use the terms "classifier" and "rule" interchangeably.

about its environment -- the system's sensory organs -- and effectors that allow it to interact with its environment. The detectors function much like the coding does in a genetic algorithm. In particular, the detectors decode environmental stimuli into messages of the type that the system can understand.¹⁸ These messages are then metaphorically posted on a message board.¹⁹ The condition parts of the existing classifiers in the system are then compared with the posted messages, to see if any of the messages satisfy any classifier's condition. All classifiers whose condition portions are satisfied are activated. In a very simple system with only a few rules it might be possible for every classifier activated to post its message; but in a complex system with an extremely large number of rules this will not be possible. The message board can only contain a fixed number of messages at any one time, and in a complex system the number of activated rules will be far greater than the available space. Thus, from an expansive list of possible rules in the particular set of conditions, the system needs a way to select some of those rules to post their messages. This is achieved in classifier systems by forcing the rules to compete with each other for the "right" to post their messages.

"All classifiers are created equal". Initially, a system begins with randomly generated classifiers, each of which is assigned a number to represent its strength. All classifiers begin with the same strength value. When a classifier is activated by having its condition part satisfied by at least one of the messages on the message board, it makes a bid to have its own message posted. A classifier's bid is proportional to its strength, the specificity of its condition part -- the number of positions in the string that have a 0 or a 1 but not a #, and the number of messages activating it, referred to as its support.²⁰ The strength of a classifier is a measure of its past usefulness to the system, with the competition favouring the stronger or previously more useful rules. The specificity of its condi-

¹⁸ It is important to note that this process is far from straightforward.

¹⁹ Such systems are sometimes called blackboard systems.

²⁰ This arrangement for determining the strength of a bid follows Holland et al, pp.116-7. Goldberg 1989, pp. 225-9 develops the notion slightly differently.

tion part corresponds to its general usefulness. Its support corresponds to how compatible a rule is with the other information active in the system. The competition between rules to have their messages posted is designed so that previously useful rules that very completely satisfy the current conditions, compatibly with the current information active in the system, have the best chance of winning. (Holland et al 1987, pp.47-9) Each rule is given a chance of winning in the ratio of its bid to the total bid of all rules in the competition, with the winners being probabilistically determined. The strongest rule does not always win, but it wins most often. For example, if three classifiers in competition made bids of 2,3, and 5, the first rule would have a twenty percent chance of winning, the second a thirty percent chance, and the third a fifty percent chance.

1.2.2 The Bucket Brigade: Paying Debts and Being Rewarded

The competition between rules is essentially random when the system first begins interacting with the environment, because all of the rules have equal strength. The system uses a method of apportioning credit to change the strengths of its rules as it receives feedback from its interaction with the environment.²¹ This results in non-random action very quickly, because the system learns from its actions. There are many different methods of apportioning credit, but here we will consider only one: the bucket brigade. According to this method, no classifier can post its message for free. When a classifier wins its competition it must pay out some fixed ratio of its strength to all of those rules that activated it, to be shared equally among them. If it was a message from the environment that activated the rule, its payment goes to the environment.²² A winning classifier will, in turn, receive payment from all the rules it activates that are successful in posting their messages. Any rule whose message activates one of the effectors rather than another

²¹ It is here that I am assuming that a system's actions can be rewarded by the environment. A possible mechanism for this is discussed in detail in section 2.3.

²² Payment to the environment for a theoretical system is simply reducing the strength of the appropriate rule. In a practical system, however, this payment could be realized as a decrease in energy necessary to make the detectors function.

rule, receives its reward²³ directly from the environment. A rule's strength changes by the difference between what it pays out and what it receives in payment. This can be a net gain or loss depending on the strengths of the rules involved, the number of rules sharing a payment, and the size of the reward from the environment (which can be negative).

Let us see how this works with an example. For the sake of clarity, I will consider a system that is sufficiently simple to post the message of any classifier that is activated. All rules will begin with a strength of 100, and payments will be ten percent of the rule's strength. The number after a message on the message board will indicate which classifier posted the message, similarly the number after a reward value will indicate which classifier paid for the reward, with (env) meaning environment in both cases. A star beside a classifier means that it is activated in that time interval. The message of classifier (5) activates an effector.

<u>Message Board</u>	<u>Classifiers</u>	<u>Strength</u>	<u>Payment</u>	<u>Reward</u>
t=0 11110 (env)	1) #11##:00001 *	100	10	
	2) 00#11:10101	100		
	3) #1#10:00011 *	100	10	
	4) 0#011:10110	100		
	5) 101##:11001	100		
t=1 00001 (1)	1) #11##:00001	90		---
00011 (3)	2) 00#11:10101 *	100	10	
	3) #1#10:00011	90		20(2,4)
	4) 0#011:10110 *	100	10	
	5) 101##:11001	100		
t=2 10101 (2)	1) #11##:00001	90		
10110 (4)	2) 00#11:10101	90		5(5)
	3) #1#10:00011	110		
	4) 0#011:10110	90		5(5)
	5) 101##:11001 *	100	10	
t=3 11001 (5)	1) #11##:00001 *	90	9	
01100 (env)	2) 00#11:10101	95		
	3) #1#10:00011	110		
	4) 0#011:10110	95		
	5) 101##:11001	90		50 (env)
t=4 00001 (1)	1) #11##:00001	81		
	2) 00#11:10101	95		
	3) #1#10:00011	110		
	4) 0#011:10110	95		
	5) 101##:11001	140		

²³ The notion of environmental reward is discussed at length in section 2.3.

In this example, the process stops at $t=4$ because none of the rules are activated by the only message on the message board. Of course, activity will resume again when a new message from the environment is posted on the message board by the detectors. Classifier (1) ended up with a much lower strength than it started with because it was unable to activate any rules, hence it received no reward, yet it had to pay to post its message. Classifiers (2) and (4) suffered from the fact that they both activated the same rule, (5), and so they had to share the available reward. Notice that over many time intervals, the large reward from the environment enjoyed by rule (5) would eventually be shared throughout the system. For example, the next instance in which rules (2) and (4) both activated rule (5), they would receive a larger reward, even though they would have to share it, because ten percent of the strength of rule (5) is larger. In this way, environmental reward is back-propagated through the system, eventually strengthening all of the rules that led to the beneficial action. This is the way in which a classifier system learns from its actions.

We will consider the effect of this method of apportioning credit in detail in the next chapter; but, briefly what the bucket brigade does is it creates a sequence of rules, "middlemen" between the messages caused by the environment and the messages resulting in favourable actions by the effectors in those conditions. The message of one rule in the sequence satisfies the conditions of the next, thereby activating it. Each time the sequence occurs, the reward from the environment strengthens an earlier rule in the sequence, until eventually every rule in the sequence is strengthened. This makes it increasingly likely that all of the rules will win their individual competitions to post their messages, thus increasing the likelihood that the entire sequence will be activated. The "middlemen" are like the members of the bucket brigades in the old method of putting out fires, because each receives reward from one neighbour and passes it along to the next, the reward being like the buckets getting passed.

1.2.3 The GA: Generating New Classifiers

It might occur that a classifier system has no useful rules in certain conditions, or worse no rules at all. In these cases the system needs a way to produce new rules. Since

rules are just strings of a constant length, they can be operated on by the genetic operators we considered above, namely reproduction, crossover, and mutation. A classifier system must perform in an arbitrary environment that is constantly presenting novel situations. As a result, it needs a large number of co-adapted rules that will lead to favourable actions in the various conditions the system is likely to encounter. As we presented GA's earlier, they were designed to be problem specific, relying on a notion of fitness that depended on a problem specific evaluator function. They were designed to find a single optimal solution for a particular problem. As we saw, however, in the course of finding one very fit string to solve the problem, an entire population of relatively fit strings is produced. This feature of GA's makes it possible to use them in a classifier system.

A classifier system has no explicit evaluator function because there is no one particular problem it is trying to solve. The usefulness of its rules is based on their past performance, which is measured by their strength. However, a GA can use the strength of the classifier system's rules as a value of fitness. When a classifier system encounters a message that either does not satisfy the condition part of any of its rules, or only satisfies the condition part of rules that result in negative environmental reward, the system can use a GA to produce new rules. The roulette wheel reproduction operator can be used to determine which rules will be allowed to reproduce and how often, with the strongest rules having the best chance. Once a parent population is selected, its members are paired and crossover is performed. Then, as with all GA's, each position of every string resulting from crossover is tested for possible mutations. Of course, the mutation operator must be modified to allow for the # character in the condition part of a classifier. When all mutations have occurred, a new set of rules is available to the system. Because the system has no explicit evaluator function, the new rules cannot be tested for fitness. Instead, they are each assigned the same strength value. The real evaluation of the rule will occur when it is activated. A useful rule to the system will result in reward, and hence be strengthened by the bucket brigade. The initial strength assignment just serves to give a new rule an opportunity to "show what it can do".

There are a few differences between a GA in a classifier system, and a GA being

used for optimization. We have already considered the fact that a GA in a classifier system has no explicit evaluator function; rather, it uses the strength of a rule, as determined by the environment's effect on the system in reaction to the actions that rule is involved in producing. Another important difference is that in a classifier system, replacement of rules must be very conservative. The system must maintain a high level of performance while it is searching for even better rules to govern its actions. Unlike an optimizing GA that replaces the entire population every generation, a GA in a classifier system can only replace a small number of rules at any one time. New rules replace only the weakest members of the existing classifier population. A further constraint can be placed on replacement so that a new rule replaces the weakest rule to which it is most similar in a bit by bit comparison. In this case the new rule fills in where it is most needed and does not replace a rule whose job it cannot do -- i.e., it must be possible to activate a new rule in much the same conditions as the rule it is replacing; the idea being that it is better to have a weak rule than no rule at all. In all other respects the GA in a classifier system is the same as the GA's we described earlier. We now consider the cognitive features of a classifier system.

CHAPTER 2

A COGNITIVE LOOK AT C1

Let us imagine a creature C1, with a classifier system for a mind. What kinds of cognitive functions does it have? What is it capable of doing and in what important ways does it fall short of having the rich mental life that humans and many animals enjoy? The classifier system will provide the basic framework for my model; C1 is my evolutionary amoeba. We will proceed by considering what cognitive features it has, and develop new systems by adding on structures to produce aspects of cognition it does not have. Since C1 qua classifier system is a machine learning system, we will concentrate on its learning abilities. This will lead us to a discussion of the nature of its representations and its brain plasticity. C1 is deceptively simple, and some quite sophisticated features are entailed by its basic structure. As we explore the nature of C1's mental universe, these features will become explicit, suggesting a natural direction for the development of other, more cognitively endowed, systems.

2.1 Learning: Slowly but Surely

2.1.1 Coupling and Sequence Formation

We saw briefly in the last chapter that a sequence of classifiers might be activated between an initial message from the detectors and an action taken by the effectors. Sequences are not predetermined by the system, but they arise when none of the winning rules activated by the messages posted by the detectors activate the effectors directly. It is also possible that none of the messages posted by the initially activated classifiers in turn activates a classifier that determines an effector action, and so on. Clearly the longer it takes before an effector is activated, the longer the sequence of rules will be. At each step in the sequence, messages that are posted satisfy the conditions of other classifiers, thereby activating them. These rules then enter into a competition with each other to post their own messages, and the winners become part of the sequence. Any rule that successfully posts its message is dependent on the rule or rules that activated it; it "owes" its success to them -- and pays them for it. The activation of one classifier by another one is a key process in the working of a classifier system, and the special relationship between

two such rules is called "coupling".²⁴ That is to say, two rules are coupled if the condition part of one matches the message part of the other, in which case the latter would activate the former.

It follows from the definition of coupling that a sequence of rules, beginning with a message from the detectors and ending with an action by the effectors, consists entirely of rules that are coupled with rules in their temporally adjacent stages, where a stage in the sequence is a single posting of messages on the message board. However, the fact that two rules are coupled is no guarantee that when the message of the first rule activates the second rule, the second rule will be successful in posting its message. Thus, a sequence that occurs in C1's classifier system brain after a certain set of messages from the detectors will not necessarily occur again, even if the messages from the detectors are exactly the same. That is to say, when C1 repeatedly finds itself in a certain situation it will not necessarily do the same thing each time. This kind of novelty is, of course, very often useful. If C1 encounters a predator from which it has once escaped, say a human, notorious for predicting what creatures will do next and using that to capture them, it would be a great advantage to C1 not to attempt escape in exactly the same way. But there are many instances in which it would be best for C1 to do exactly the same thing it did the last time, particularly if the action led to significant environmental reward, like taking shelter in a storm or climbing a tree for food. The bucket brigade method of apportioning credit allows C1 to learn from the good actions it happens to discover.

C1 has no memory apart from its explicit set of rules with their variable strengths; so it cannot just remember good actions in certain conditions and perform them when appropriate. It must modify rule strength in such a way that similar conditions result in the same rewarded action. It does so as follows. When a sequence of classifiers leads to an action that the environment rewards, the final rule in that sequence is greatly strengthened by the reward. It might take several encounters with the same conditions before C1 produces the same sequence, but when it does, the second to last rule in the

²⁴ This term is defined in Holland et al, p.50.

sequence coupled to the one previously rewarded will share in the benefits of the reward. This occurs because with its increased strength, the last rule is much more likely to win its competition to post its message, and when it wins it must pay the previous rule that activated it. The effect is that the mere activation of the earlier rule is likely to result in both rules successfully posting their messages, because each is stronger than its competitors. A third instance of the sequence will spread the reward back to the third last member in the sequence, and of course, the later rules continue to get stronger as long as the action leads to environmental reward. As the reward spreads to earlier and earlier stages in the sequence it becomes increasingly more likely that the entire sequence will occur once it begins. The reason is that once a strengthened rule is activated, it is very likely to win its competition, as are all the rules following it which have also been strengthened, some of them many times; and each occurrence of the sequence strengthens a rule one stage earlier. Eventually, every rule in the sequence is strengthened, increasing the likelihood that the entire sequence will occur in the appropriate conditions, i.e., when the detectors post the messages that activate the first rule in the sequence. Once the entire sequence has been strengthened sufficiently, C1 has learned to perform the action the sequence prescribes under the conditions that activate the sequence.

Sequences are serial in nature, containing only one classifier at any stage. At a single stage, however, there are many messages posted on the message board. Thus, C1 produces many sequences in parallel with each other. Since it is possible for one rule to be coupled to more than one other rule, a single message can activate two or more rules; i.e., some sequences can split at a certain stage. In our example from the previous chapter, showing how the bucket brigade works, splitting occurs at stage $t=1$ when rule (3) activates rules (2) and (4). It is also possible for sequences to merge, since two distinct messages can satisfy the condition part of the same classifier. This occurs in our example at stage $t=2$ when both rule (2) and rule (4) activate rule (5). Both of these situations result in a single classifier being a member of more than one sequence. Parallel sequences are important to the system because they can be coordinated to produce an action involving more than one effector by being initiated under the same conditions. This

allows C1 to integrate its effector movements. In the next chapter, we will see how C2 greatly improves this ability.

The sequences that occur in our example are: env,1; env,3,2,5, action; env,3,4,5, action; and a second instance of env,1, with a different message posted by the detectors from the first instance. The sequences ending with rule (1) do not lead to any action by the system, and hence receive no reward. As a result rule (1) is continually weakened. In a system with competition, a rule such as rule (1) will eventually become so weak that it cannot successfully post its message. In this way C1 does not waste its limited cognitive resources exploring useless options. Eventually, rule (1) will be replaced by a new rule produced by the genetic algorithm. C1 is very good at learning from its mistakes.

The sequences, env,3,2,5, action, and env,3,4,5, action provide an example of a single sequence splitting into two sequences, and then merging back into a single sequence. The sequences coincide at all stages but one, both containing rules (3) and (5). This redundancy may seem inefficient, at first glance, but it is valuable to C1. If C1 is injured in some way that destroys rule (2), for example, the useful action activated by rule (5) leading to environmental reward can still be performed. So C1 is able to partially protect itself against brain damage. All is not lost in the case of a minor injury. Moreover, C1 can use its redundant discovery of alternate "neural" routes producing the same action to learn more efficiently. In particular, since the bucket brigade method of apportioning credit back-propagates reward by only one member of a sequence each time the sequence occurs, the sequence must be produced at least a number of times equal to its length before it can be learned. Now if C1 discovers two different sequences that activate the same action in the effectors and one of the sequences is shorter than the other, it can learn to perform the prescribed action given the initial conditions much faster than if it was able to learn only the longer sequence.²⁵

²⁵ For a sequence consisting of k classifiers, with a probability p that the entire sequence will occur given the initial conditions, i.e. that every rule in the sequence will successfully post its message, the expected number, N , of instances of the initial conditions required to learn the procedure is given by $Np = k$. Clearly the number of times C1 must encounter the initial conditions in order to learn a sequence is proportional to the length of the sequence, hence shorter sequences are learned more quickly.

2.1.2 *Default Hierarchies*²⁶

One of the factors influencing a classifier's bid to post its message is its specificity, defined as the number of non-#'s in the condition part of the classifier, divided by the length of the condition part. Thus, a classifier with all of the positions of its condition part specified has specificity 1; and a classifier with only #'s in its condition part has specificity 0. (A totally unspecified classifier cannot successfully post its message. The reason for this is that it has no "idea" about what is "going on in the world". It will try to post its message under any, in fact every, set of conditions.) The competition between classifiers to post their messages is biased in favour of the more specific classifiers. The effect of this bias is to choose rules "designed" to operate in the very conditions under which the competition is being held. All of the rules activated to participate in the competition are coupled to some rule that has posted a message; but some are much more strongly coupled than others, in the sense that one rule's condition part might match a message exactly, whereas another might be so general that it matches the same message almost by default. A very general rule will match almost any message. If two rules of comparable strength are competing, one quite general and the other very specific, the specific rule is preferable because its strength must have been obtained in virtue of its usefulness in the present conditions. A general rule, on the other hand, may have been rewarded for its involvement in very different conditions, and so is more likely to lead to unfavourable actions. Of course, if both rules are quite weak, neither is likely to post its message regardless of specificity.

The bias in the competition to post messages towards specific rules improves the efficiency of learning sequences. One of the impediments to learning a sequence is that even if the initial conditions that activated the sequence recur, there is no guarantee that every classifier in the sequence will be successful in posting its message. If the sequence is interrupted, then the final classifier will not be activated and so cannot back-propagate the reward it received when the sequence did occur. However, if the rules contained in a

²⁶ This notion is developed in Holland et al 1987, pp.18-19.

sequence are very specific, they will have a greater chance of posting their messages when they are activated. This reduces the chance that the sequence will not be completed, once conditions activate its first rule, thereby improving the rate of learning.

The specificity factor in a classifier's bid to post its message induces a structure on the classifiers, referred to as a default hierarchy. Messages posted by the detectors will activate several rules. The action C1 takes will depend upon which of these rules post their messages. That is, C1 has different actions available to it in a given set of conditions. The bidding competition is arranged to favour an action that was successful in these or similar conditions in the past, by choosing the more specific rules to post their messages. A specific rule can be thought of as an exception to the more general rules.

A general rule will lead to a certain action in a broad range of circumstances. However, it might occur that in certain conditions this rule performs poorly, even though it leads to reward in many other situations. The poor performance will activate the genetic algorithm to produce new rules. When a more specific rule that leads to reward in the conditions in which the general rule performed poorly is produced, it will win the competition to post its message over the more general rule in those conditions. The general rule will not be replaced in virtue of its past successes, but it will not post its message in the conditions in which it performed poorly. Thus we have the rules organized in such a way that under a wide range of conditions, a general rule is available to guide the system's actions, but when certain exceptional conditions occur a more specific rule will take control. As an example, if C1 detects a round object coming towards it, it will generally move; but in the exceptional case that the object is white and has been propelled by a wooden stick, it will catch the baseball.

Default hierarchies can form several layers deep, with exceptions to the exceptions. Nonetheless, it is important for C1 to keep general rules along with its specific rules, even though specific rules are favoured in the bidding competition. The reason is quite simply its limited cognitive resources. Obviously the best thing for any system would be to have a very detailed sequence of rules to handle every possible situation it might face. However, the novelty of the environment makes this impossible, so systems

have to do the best they can. By having strong general rules mixed with strong specific rules, C1 can quite efficiently learn a variety of actions for different situations. Whenever possible, it will base its actions on rules specific to the conditions it is facing; but in the event that it has no specific rules that match the given conditions -- simple counting arguments guarantee that there will be such instances -- there are rules available to determine some course of action. The general rules take over by default, hence the name default hierarchies. It is important to note that a default hierarchy forms purely mechanistically. No categorical or other semantic structure is involved. The simple mechanism of including a specificity factor in a bid is all that is required to induce this structure on the system.

Relying on a default rule is not just "doing any old thing" when a specific rule is unavailable. General rules that survive are strong because they lead to reward in at least some circumstances, circumstances similar to those in which they post their messages by default. A default rule is an old standby that has always worked reliably when it was needed in the past, and can be used when nothing obviously better presents itself. In contrast, general rules that do not lead to reward are weakened very quickly, making it extremely unlikely that they will post their messages. Such rules are a nuisance for the system, however, because they take up an inordinate amount of the system's resources. Being very general, their condition part is satisfied by many messages. Thus they are frequently entering into competition to have their messages posted. It only takes a few successful postings before this type of rule becomes very weak and cannot successfully post its message again, yet it continually tries. This requires determining its bid, which because of the support factor must be calculated for each new bid, occupying the system's resources in totally unprofitable ways. What C1 includes to eliminate such rules is a tax. Every rule that bids must pay a tax from its strength just for entering the competition. The tax is a very small fixed amount, but for a rule that bids frequently with no prospect of reward, the tax significantly decreases its strength. The result is that such a rule quickly dies off; it is replaced by a new application of the genetic algorithm, and the system's resources can be used more profitably to find better rules.

2.1.3 The Effect of Sharing the Rewards²⁷

If a rule that is activated by more than one message wins the competition to post its message, its payment to the rules that activated it is divided among them. One obvious reason for sharing is that if it had to make full payment to each rule activating it and there were several such rules, its strength would be disproportionately weakened. In fact, if many rules come to have messages that activate one rule, the likelihood is that it will be a strong rule. The genetic algorithm chooses the strongest rules to reproduce, i.e. rules that have been rewarded. If these rules do not activate the effectors and so receive reward directly from the environment, they must activate other strong rules, which pay them well for doing so. The features of a parent classifier that make it strong are often passed to its offspring as building blocks. But in the case of a classifier whose message activates other classifiers, the building block would have to match the corresponding positions in the condition part of some strong rule; otherwise, it would not be responsible for the classifier's strength. The new rules to which the building block is passed on will also inherit other characteristics from their parents, making it likely that both parent and child will be activated at the same time, and that both will succeed in posting their messages. In this case both will activate the same strong rule. This rule is activated because it is useful to the system, and thus it is not desirable to reduce its strength by a factor of two or more simply because it was activated by two or more other rules. Sharing helps good rules stay strong by not forcing them to overpay for their success.

Of course, causing rules to share the payment made by a winning rule that they each activated has an effect on those rules. For example, when a parent classifier produces offspring that ultimately activate the same rule as the parent, as described above, the parent receives a smaller reward than it did before the offspring were members of the population, because it is forced to share the reward. This tends to weaken the parent rule, or at least decrease its rate of growing stronger. Ironically, it is the strength of the rule that determines its chances of reproducing, and reproduction that leads to sharing means

²⁷ This effect is similar to the development of species and niches in genetic algorithms. My treatment is an application of the discussion in Goldberg 1989, pp.185-192.

that its future chances of reproducing are not as great as they would have been if the rule had never reproduced in the first place. Classifiers are not allowed to be miserly about their rewards. If there is a large reward available, others will soon be along for their portion of it. If this continues too long, there will not be enough to go around. The rules will weaken and some will die off, i.e. be replaced by new strings produced by the genetic algorithm. As these strings get weaker, few if any will reproduce, so the number of strings with the building blocks necessary to share the reward will decrease. This is a classifier system's population control.

What sharing does for the system is to introduce stable subpopulations -- rules co-adapted to initiate a certain action that leads to reward -- in the manner just described. The advantage of having subpopulations is that it maintains diversity among the classifiers, which is extremely important for a creature trying to function in an ever-changing environment. If every rule received the total reward available, the population would quickly converge, yielding a set of classifiers all with the building blocks necessary to obtain the largest rewards, a phenomenon known as genetic drift. This would leave the system inadequately prepared for conditions that did not activate these rules. Sharing restricts the number of rules that will converge in such a way as to receive the same reward. The number of classifiers that can exist in a stable subpopulation is proportional to the size of the reward. Essentially what C1 does by forcing its rules to share, is to allocate a number of rules toward obtaining a reward in proportion to the size of the reward. Since most of the classifiers receive their rewards as payment from other classifiers, sharing couples rules, hence forming sequences, in appropriate number for the environmental reward obtained by the effector actions prescribed by those sequences. That is to say, sharing provides C1 with a set of co-adapted rules. Put another way, sharing regulates how much of its cognitive resources C1 will use in learning any one thing.

2.1.4 Plasticity: The Substance of Learning²⁸

The only procedures hard-wired into C1 are those meta-rules that make it a

²⁸ The notion of learning I am considering is that of modifying behaviour.

classifier system, also referred to as the operating system. For example, the bit by bit comparison mechanism necessary to determine which classifiers are to be activated by the message list; the mechanism for determining each classifier's bid; the genetic operators, reproduction, crossover, and mutation; and, the bucket brigade readjustment of classifiers' strengths, to name some of them. These procedures are unalterable in C1. Nothing it experiences will cause it to alter these procedures in any way, except of course damage to the system. The reason C1 can have hard-wired routines totally determined from birth, despite the novelties of the environment, is that some of its actions are essentially independent of the environment. C1 needs to be able to learn, in order to cope with its environment, but the way in which it learns can be predetermined and fixed, indeed, some way *must* be. We can consider hard-wired procedures to be rules that have no strength value and so cannot be modified. If their conditions are satisfied, the action they prescribe is taken.

Learning relies on variability, and so cannot be the result of hard-wiring. C1 manages learning by having variable rule strengths that can be modified according to the usefulness of a rule. This controlled variability is the system's plasticity. All of the "interesting" things that C1 does, its interactions with its environment, are a result of its plasticity; the procedures that it uses to learn those "interesting" interactions are themselves hard-wired. Thus we can classify the rules in C1 as one of two types: (1) Hard-wired meta-rules that control the basic functions of the system. These rules never change and always act when their conditions are met. (2) Rules of variable strength that must compete to participate in influencing the system's actions; their strength is modified depending on the performance of the rule as judged ultimately by environmental reward or lack thereof. The role these rules play in the system is constantly changing as the system adapts to the novelties of its environment. Rules that perform poorly can even be replaced by newly generated rules, with replacement being guided by the best rules currently in the system. This second class of rules constitutes the system's plasticity.

As our C1 system evolves, greater use will be made of plasticity so that more evolved systems not only have the capacity to learn, but can learn better how to learn. We

will keep the basic distinction between plasticity and hard-wiring, as defined here, throughout. That is to say, all of our systems will incorporate hard-wired system governing rules, which have no strengths because they are always prescriptive when activated, and variable strength rules, constituting plasticity, that compete for control. Hard-wiring controls how the system works, which does not change, and plasticity controls what the system does, which depends on the circumstances. Plasticity will be very important for developing an account of learning, particularly single-instance learning -- i.e., learning from a single encounter with conditions in the environment -- like our own, though all learning depends on it equally.

2.1.5 General Features of C1 Learning

One of the most striking features of C1 is that it is not too quick. Of course, it can learn a simple sequence rather easily. For example, if it has a rule that is activated by some message from the detectors whose own message activates the effectors, then a single instance of that rule winning and being rewarded can greatly increase its strength. The next time the detectors post that message, the newly strengthened rule will be very likely to activate the effectors and if it is rewarded again, its future success is virtually guaranteed. However, if a sequence that C1 is trying to learn is long, and many rules compete to post their messages at each stage, strengthening the entire sequence will take many exposures to the initial conditions -- the set of messages posted by the detectors. In fact, for a sequence involving k classifiers, strengthening the sequence will require at least k instances of the initial messages, the number being much higher if some of the rules in the sequence do not manage to post their messages. Even if we take an example very favourable to C1, in which a sequence consists of ten classifiers and each of them is so strong that it has a ninety percent chance of posting its message when activated, there is only a thirty-five percent chance that the entire sequence will occur when the initial messages are posted. It follows that there are only two chances in 100,000 that the entire sequence will be strengthened in the minimum ten occurrences of the initial conditions. In fact, we would expect C1 to need approximately twenty-nine experiences of the initial conditions to learn such a sequence, and there is no guarantee that C1 will ever learn it.

I mentioned earlier that C1 was quick to learn from its mistakes, and in the case of our simple example it was. In fact, it will be quick to learn that any rule not able to activate other rules in the system or the effectors is utterly useless, a cognitive dead-end. Such rules will quickly die off. However, C1 is not fast to learn from its mistakes in situations in which a sequence leads to some action that produces negative results in the environment, especially if the sequence is a long one. The trouble is that the rule leading to the unfavourable action might be many stages removed from the effectors. It might well be the case that the sequence following the offending rule can lead to reward under the right circumstances, but this rule has activated it at an inappropriate time. To undo this correlation, the rules have to be weakened back to the offending rule one stage at a time, just as reward is back-propagated. As we have just seen, this can be a long procedure. It also has the undesirable side-affect that potentially useful rules are weakened and will have to be relearned.

Both in the case of being a slow learner and the case of being unable to quickly detect the source of an unfavourable action, C1 suffers from not having a model of its own inner workings. As C1 evolves it will have to develop a model of its models. This will be an important step in giving an account of single-instance learning.

2.2 Representations: How C1 "Sees" the World

2.2.1 What Constitutes a Representation

Intuitively, for many representations, if R is a representation of x, then R is about x, R conveys some information about x. A representation is something that can stand in the place of something else because it is about that other thing. Of course, the notions of "stand in the place of" and "about" need to be made precise. A Shakespearean sonnet can be about love, but it is not immediately clear in what sense the poem could "stand in the place of" love. In the context of a cognitive system, the representation R can be thought to "stand in the place of" x if it allows the system to engage in the same cognitive activities that it could perform in virtue of an actual encounter with x. Being able to form representations is a great advantage to a cognitive system because it allows the system to make decisions about things even when they are not present. Based on a representation of

x , a system can determine a course of action concerning x regardless of whether or not x is actually there. Though far from a sufficient condition, it is essential for any system modelling human cognition that it be adept at forming and manipulating representations. We regularly engage in activities like imagining the taste of banana cream pie, adding numbers in our head, and remembering a beautiful sunset. Since in none of those situations is the object of the cognitive activity actually present, it is only by having a representation of the object that we can engage in those activities. In particular, it is only because we have a representation of x that our cognitive activities can be about x .

The "aboutness" aspect of a representation is a defining characteristic. If something is not about anything, regardless of what that "anything" might be, it is not a representation. Someone who asserted that R was a representation, but responded to questions about what R was a representation of with confusion or by stating that it was not a representation of anything could not be said to understand the meaning of the word "representation". To be a representation just is to be to be a representation of something. Two features of a representation taken together comprise its "aboutness": reference and sense. For R to represent x it must refer to x ; R must indicate what it is representing. Reference is like a naming function; it is a feature of a representation that it "points" to the thing it is representing. In addition to indicating what is represented, R must convey some information about it. A representation includes something like a description of the thing it refers to. This information is the content of the representation; it depicts x in some way, and this depiction influences what action the system forming the representation takes. Neither reference nor sense alone is adequate to make a representation "about" something, because the former is devoid of content and so has nothing to say about what it refers to, and the latter contains information out of any context; there is no indication of what the information pertains to. With both, however, R contains information about x , the thing referred to, and so is about x .

The third essential feature of a representation is that it represents the information about x to someone or something. This third feature has the most interesting implications for modelling human cognition because it requires a representation to have a subject,

though the nature of this subject may be very different from our introspectible homunculus. "If there were intrinsic representations -- that is to say, states that represented, and represented one specific thing, no matter where they were, how they were connected to other representations, or whether they were related to minds -- then this would not be the case; representations would not have to represent to someone [or something] ... But there is a well-known, compelling reason to deny that anything can be an intrinsic representation in this sense; anything can represent more than one thing in more than one way, depending on how it is taken. A 'taking' requires a 'taker'" (Brook 1994, pp.26-7). For example, an arbitrary sequence of 0's and 1's, independent of a key and an algorithm, does not constitute a mathematical code. The sequence can be "correctly" interpreted in a myriad of different ways, depending on the decoder. But someone or something must determine the key and the algorithm used in decoding, and this is what fixes the content of the sequence. Without a subject there can be no representation because physical things only instantiate sense and reference if they are taken to do so by someone or something, i.e. if there is a subject.

Based on the above considerations we define a representation as follows. R is a representation of x if it satisfies three conditions: (1) R contains information that allows the system to refer to x in some way; it indicates that x is its object (reference); (2) R contains or conveys information about x, it depicts x (sense); (3) Conditions (1) and (2) can only be satisfied if they are done for someone or something (function).²⁹ "Every representation of something represents it *to someone* [or something]" (Brook 1994, p.26).

²⁹ I am defining only one type of representation here since it is the only type of representation that will be active in my later systems. It is an open question whether these system could form other types of representations such as those typically associated with natural languages. This definition is very close to Dretske's notion of a representation (Dretske 1988, p.70). Dretske treats function differently than the way I am using it here. He states that a representational system has a function to indicate the sense and reference of a representation. But this entails at least that the sense and reference of a representation are sense and reference *for* the system, which is my third condition.

Representations are about something to someone or something.³⁰ We can now apply this definition to C1.

2.2.2 C1's Lack of Representations

C1's detectors decode environmental stimuli into messages that the system can "understand". What messages the detectors actually post will depend on what they can detect and what there is in the environment to detect. It is tempting to say that a message posted by the detectors is a representation of a certain state of affairs in the environment; however, based on the definition of a representation that we have just developed, this is not the case. There is no doubt that something about the environment causes the detectors to post the messages that they do, but causation alone is not constitutive of a representation. Event A can cause event B without B referring to A or containing information specific to A for anyone or anything. A baseball bat can cause a baseball to fly 400 feet but neither the ball nor its flight represent the bat. In particular, the flight of the ball, the state caused by the bat, could have been caused in many ways, say by any other baseball bat, a cricket bat, or a pitching machine, for example. The flight of the ball alone is not enough to indicate the cause of its flight.

Events that have many causes do not refer to a particular cause in a given situation. Nothing about the event in isolation can indicate which of many possible causes was the actual cause in the particular situation, so an event does not refer to its specific cause merely by being caused.³¹ Furthermore, having many causes precludes an event from containing information specific to a particular cause solely on the grounds of that causation. There is nothing about the event itself, if it is in fact the *same* event, that

³⁰ My final model will be a realist treatment of representations, but even instrumentalists like Dennett can accept this definition. The disagreement is not so much what a representation is – though that is inevitably involved – rather the argument is about the basis on which something can be said to have a representation. Realists argue that whether representations are really there or not is a matter of fact. Whereas Dennett argues that a representation and other intentional objects are posited to predict the behaviour of something. (Dennett 1990)

³¹ Watering down the notion of reference so that an event refers to all of its potential causes will not make an event a representation of its cause. A representation R, of x must refer to x not to x or y or z.... With such a weak notion of reference it is not possible for R to refer just to x, meaning that R cannot represent just x.

makes it possible to discriminate its various causes, from which it follows that the information the event conveys applies to all of its possible causes and does not depict a particular cause in a given situation. Hence an event with multiple causes cannot refer to a particular cause nor convey information of a particular cause solely in virtue of that causation. An event with several causes cannot be about any one of its causes simply because of the causal interaction, which implies that the event cannot be a representation of any one of its causes merely in virtue of that cause being a cause.³²

Now in the case of C1 -- in fact, in all of the systems we will consider and in humans also -- the messages posted by its detectors can have many different causes because the system has only finite discriminatory capabilities. That is to say, the system cannot differentiate all of the states of its novel environment. Some states of the environment must be detected in the same way, i.e. cause the same detector messages to be posted. Also, in the case of C1, there is no connection, other than causation, between environmental events and the messages they cause to be posted by the detectors. It follows that the messages in the detectors cannot be representations of the events that cause them to be posted, since, as we saw above, causation alone is insufficient for determining a representation.³³

It is important to stress that despite the fact that the messages posted by C1's detectors are not representations of events in the environment, they do contain information about the environment. A detector operates by transforming patterns in the environment into messages in a lawful way. What messages are posted by the detectors is determined exclusively by the detectable patterns in the environment and the structure of the detectors. Thus, some state of affairs in the environment causes a certain state of affairs in the detectors according to physical laws, by which the states are related. In virtue of these laws, a detector message is a transformation of some environmental pattern of, say,

³² Note that I am not arguing that an event can never be a representation of one of its causes. My point is simply that something in addition to the causation is required.

³³ See section 4.1.2 for a discussion of Fodor's notion of asymmetrical dependence as a solution to problems of multiple causation.

light or sound. Now, these environmental patterns carry information about the environment, so that when a detector transforms them, it transforms the information into the messages that it posts. Since transformed information is still information, the messages in the detectors contain information about the environment.

Other messages in C1, those not posted by the detectors, are not representations either, nor do they contain information about other messages in the system. Non-detector messages are not representations because they are only in causal relation to other rules, which is insufficient to determine a representation, as we saw above. Also, the values of the bit positions in the message part of a classifier are not directly related to anything occurring in the system before the message is posted, including its condition part. Thus, the message does not convey information about messages activating the classifier in which it is contained. Unlike the messages posted by detectors whose bit position values are determined by the environment, the bit position values of other messages in C1 are independent of the messages causing them to be posted and so contain no information about those messages. This means that information about the environment is detected and posted by the system, thereby playing a causal role in the system's actions, but not in any direct way since that information is not passed through the system to its effectors.

The condition parts of classifiers fare no better than the message parts as candidates for being representations. Since the condition part of rule is activated by messages that match it, the condition actually contains information about those messages, namely that they have certain specific values in bit positions matching the value of the corresponding specified positions in the conditions. Furthermore, this is information to the rules in the operating system that determine which messages will participate in the competition to be posted, since the match alone guarantees that the message will compete. Thus the condition part of a classifier satisfies the second and third conditions in the definition of a representation. However, the condition has no way of indicating the message its information is about. If a condition is totally specified then it is entirely information about the single message that matches it and so has no way to refer to that message. Otherwise, it is matched by several messages that only differ where the condition has wild card

characters, but in this case the specified bits again convey information about the messages that match the condition, and the wild card characters do not convey any information at all. So in neither situation does the condition refer to anything, which precludes it from being a representation. Now since the only things in C1's cognitive architecture are its rules, conditions and messages, and these are not representations, C1 does not have any representations.

2.2.3 An Instrumentalist Methodological Concern

The above method of determining whether or not C1 has any representations is one that an instrumentalist like Dennett would flatly reject. Our approach to determining what representations C1 has is to consider what in C1's cognitive architecture -- which is nothing but classifiers in the case of C1 -- could be a representation; i.e., our approach is a realist one. We assume that unless something that is "really there" in C1 is a representation, it does not have any representations. Dennett's method of determining whether or not C1 has certain representations, or any other intentional states, is to use the intentional stance. The Dennettian story goes something like this. Briefly, the intentional stance is a predictive strategy in which we posit the existence of intentional states in a system in order to make sense of, and hence predict, its behaviour. Whether or not the system "really" has that intentional state is not something for which there are criteria. If positing an intentional state to the system makes it possible to accurately predict its behaviour, then the system is in that state (Dennett 1987). So the way to determine whether or not C1 has any representations is not to look inside it, but to see what it does. If positing representations makes it possible to predict C1's behaviour then it has them, in the same way that we do.

Interestingly enough, despite disagreeing on the method for determining if C1 has representations, Dennett would not disagree with the conclusion that it does not. The reason is that the existence of some intentional state does not rely on whether or not we *could* predict a system's behaviour by positing that state, but whether or not we *must* posit that state, i.e., whether or not we gain any predictive advantage by doing so. This is made clear by the following example. It is perfectly reasonable for Captain Ahab, in

Melville's *Moby Dick*, to adopt the intentional stance towards the white whale. A whale is the kind of system that behaves in such complicated ways that the only way for us, with our current understanding, to make sense of its behaviour is to ascribe intentional states to it. On my reading of *Moby Dick*, I take it that Ahab ascribed the wrong intentional states to the whale, that it was not a vindictive hater of sailing vessels and their crews. The tragedy of the story is that Ahab made an error any of us could make. But what if Ahab had lost his leg because it had been crushed by a tree falling on it in the forest? Suppose Ahab became convinced that the tree was evil and had to be destroyed. Rather than forming a lumberjack posse to help him destroy a monster, people would think him mad. Trees do not have intentional states; they do not hate, or desire, or believe anything. Nothing about the way a tree acts, if that is even the correct description, requires ascribing intentional states to it, and so even from the intentional stance it does not have them. *Moby Tree* will not join the ranks of the classics.

Ahab could predict the "actions" of a tree by referring to its beliefs and desires; also, it is possible to predict the movements of a tornado by positing a hatred of trailer parks to it; but in both cases there are better ways to make predictions. Any intentional talk in these situations is purely metaphorical. We could describe a thermostat as "wanting" to keep the house warm, and "believing" that to do so it must activate the furnace when it "represents" the room as too cold. We could also describe a thermostat as a strip of metal that contracts when it gets cold, operating a switch that activates the furnace. The existence of a simple non-intentional description of the thermostat's actions -- as simple as the intentional one -- precludes the need for positing intentional states to the thermostat. Even from the intentional stance the mere possibility of prediction is not sufficient to posit the existence of intentional states. Some predictive advantage must be conferred before the intentional states are posited as existing.³⁴ Tornadoes and trees do not hate things from

³⁴ This account makes the existence of intentional states dependent on "who is doing the predicting", but I am only making explicit what is already implicit in Dennett. Being able to predict the behaviour of something requires a predictor, and as we will see in chapter 4, this is the weakness of the instrumentalist position.

the point of view of the intentional stance since no predictive advantage is gained by assuming that they do. Now what about C1? There is a complete account of its actions available in terms of its rules. No predictive advantage is gained by positing intentional states to C1. Therefore, it is unnecessary to ascribe representations of events in the environment to C1, meaning that even from the intentional stance it does not have them.

It is interesting that the quite different methodologies of realism and instrumentalism result in the same conclusion that C1 does not possess any representations. Nonetheless the methodological differences between the two positions are important, particularly where more complex systems are involved and our intuitions are less clear about what is going on. We will maintain a realist stance throughout our development of different systems. Our methodological differences with instrumentalism become acute in the system C3, since they lead to entirely different accounts of semantic content in the system. We will consider the shortcomings of the instrumentalist methodology in that context (4.1.2).

2.3 Rewards

All of the systems we will consider must be able to receive reward from the environment. It is because of environmental reward that rule strength can be modified, which determines which rules are active in the system, which rules reproduce, and even which rules are replaced. In short, the entire cognitive architecture is influenced by environmental reward. Therefore, it is essential that we identify some possible mechanism by which reward can be assigned; the process of reward cannot depend on some kind of intelligent assignment of costs and benefits, since that would reintroduce a homunculus into the model and undermine our efforts.

Systems themselves contain the essential mechanism for reward in their plasticity. In our model, plasticity is simply variability of rule strength. The operating system contains within it a procedure for strengthening or weakening rules. Part of this procedure consists of the bucket brigade and the tax system for competing to post a message. In these instances the amount a rule's strength is to be altered is algorithmically determined and so easily realized by mechanical processes. Now in addition to these strength altering

processes, the environment can effect the system in such a way that rules are strengthened or weakened also. The simple picture is that the environment rewards the system by strengthening all rules that are active when some action is taken, provided that action is "useful" to the system, where "useful" is meant to be unpacked along the lines of "confers survival advantage". Of course, this simple picture is woefully inadequate to account for human interaction with the environment, but then so is every other existing mechanical model, so before rejecting it out of hand, let's see how it gets off the ground at all.

When a system interacts with the environment, the environment causes certain physical changes in the system, full stop. The changes that concern us are those that could have a cognitive influence on the system. Now apart from direct manipulation of the cognitive hardware, which does not occur in the normal functioning of the system, the only influence the environment has on the system's cognitive activities is that it causes certain messages to be posted by the detectors. So any reward that the environment is to provide the system must be related to the messages posted by the detectors. Since messages posted by the detectors contain information, a simple relation could be that some messages from the environment come with information about rewards. That is to say, a message posted by a detector initiates the system's rule altering procedure. Some messages could contain information that causes the system to strengthen the active rules when an action is taken, some to weaken them, and others would not initiate the rule strengthening procedure at all, with different messages causing different magnitudes of strength alteration. Environmental rule strengthening could serve as a mechanistic analogue of pleasure, whereas rule weakening could serve for pain.

Clearly the difficulty with this simple picture is that the correlation of actions that are in the system's best interests with this mechanism of reward seems to be nothing short of a miracle. Perhaps worse, it could be the result of a homunculus in the rule strengthening procedure, one that knows what is good for the system and rewards it appropriately. Things may not be as bad as they seem, however. This simple picture is an instance of the Law of Effect, which says that stimulus-response pairs are selected for their "usefulness". In our case, the stimulus is in the environment and the response is performed by

the system. The question is how are pairs of responses selected, i.e., on what grounds are they deemed "useful". "We already have a model of a theory that admirably discharges just this *sort* of burden in the Darwinian theory of evolution by natural selection, and as many commentators have pointed out, the Law of Effect is closely analogous to the principle of natural selection. The Law of Effect presumes there to be a "population" of stimulus-response pairs, more or less randomly or in any case arbitrarily mated, and from this large and varied pool, reinforcers *select* the well-designed, the adaptive, the fortuitously appropriate pairs in an entirely mechanical way: their recurrence is made more probable, while their maladaptive or merely neutral brethren suffer "extinction", not by being *killed* ... but by *failing to reproduce*" (Dennett 1978, p.73).

According to the Law of Effect, "useful" actions by the system are those that promote its survival. How far can we get with such a notion? Our simple picture is that systems have a mechanism for altering rule strength and that messages from the environment can activate that system so as to strengthen or weaken rules by an amount depending on the message. This much at least can be easily realized from the mechanisms we have considered. Messages activate the rule strengthening procedure by matching conditions in the rules of that procedure and the amount of increase or decrease depends on exactly which rules in that procedure are activated. Now, at least initially, there is no correlation to what messages from the environment lead to rule strengthening or weakening; it is totally random, though fixed for a given system because its operating system rules are hard-wired. Some systems will reward the detector-effects of actions that are harmful to the creature, others will reward the detector-effects of neutral actions, and a fortunate few will reward the detector-effects of actions that help the system survive. A system that rewards eating rocks will not live long enough to reproduce, so the mechanism linking rock-eating with reward will quickly be eliminated from the population. Contrast this with a system that rewards eating bananas. It will flourish and pass the link between banana eating and reward to its offspring. In fact, passing links between certain actions and rewards is a very efficient way of providing offspring with what they need to survive. It disposes them to learn the actions their parents performed for survival without hard-wiring

in the procedures and thereby reducing adaptability.

The connection between survival and reward can at least result in systems with a basic correlation between actions in their best interests and rewards. Included in actions that systems will reward would be eating certain foods, reproducing, and avoiding predators. Now nature has had three billion years to develop these themes, so it could produce increasingly complex systems that are rewarded for increasingly subtle differences that lead to survival. For example, systems that reward actions of cooperation with conspecifics are more likely to survive than systems that must fend for themselves. Another example: predators face a system when they attack it and thus present a symmetrical appearance; the act of recognizing symmetry can come to be rewarded since this helps systems escape from their predators. (Obviously a very advanced system is required for this; C4 is such a system.)

An interesting consequence of the fact that certain information in the system will always lead to reward is that many actions can come to be rewarded that do not confer immediate survival advantages. For example, a system rewarded for recognizing symmetry will be rewarded even if that symmetry appears in a garden. This system has an "aesthetic appreciation" of symmetry because its ancestors survived by being able to recognize predators. Thus, merely having a procedure by which detector messages cause rules active when an action was taken to be rewarded can result in many actions being rewarded.

More evolved systems can develop even richer message/reward correlations. Many actions that are entirely neutral to the system's survival can be strengthened or weakened because of messages from the environment. There is no reason why only actions directly related to survival affect the environment in such a way that it posts messages altering rule strength. Messages resulting from survival-neutral actions could also cause rule strengths to be altered.³⁵ Since nothing hangs on how the rules are altered, randomness

³⁵ There is some problem with how much rule strength would be affected by these rules. This is often resolved in terms of the system's goals but that is simply question-begging. It is possible that amount of rule alteration itself is randomly set throughout the population, so that messages that are strongly rewarding for

concerning these actions would persist throughout the population, but the particular correlations would be important for an individual system. What actions it takes would be greatly influenced by which rules were strengthened and which were weakened. Thus, if we take "usefulness" to mean "leading to environmental reward", actions directly related to survival are useful to all conspecific systems, but individuals can vary with regard to what other actions are useful. Such a notion might eventually be developed into the mechanistic underpinnings of desires and aversions.

We have seen that our simple picture can actually lead to a quite developed mechanistic account of environmental reward. Nonetheless it is still far from adequate. Complex systems such as humans have enormous numbers of messages being posted by their detectors at any given time. Does the system add all of the reward prescribed by each rule and share this equally among every rule active when an action was taken? Are detectors and effectors locally connected so that reward is distributed locally? How can a system reward actions when the reward is not received until long after the action has been taken, sometimes years in the case of humans (as anyone writing a thesis knows)?³⁶ These are difficult questions for which I have no adequate solutions. In this I am in good company since the influence of the environment on cognition remains an unsolved problem in cognitive science, hence well beyond the scope of this paper. However, since we were able to at least get a foothold on a mechanistic account of reward, there is some cause for optimism that a solution can be found. For the remainder of this project I will operate on the assumption that solution will eventually be forthcoming and so treat environmental reward as if it is non-problematic.

For purely mechanistic reasons a system is always disposed to act so as to obtain the greatest reward available in a given situation. This is an immediate corollary of the

some individuals are merely neutral or even negatively rewarding in other individuals. Goals would then be derivative on this more basic notion of reward.

³⁶ The problem of long-term reward is perhaps the most insidious. My intuitions about its resolution are that very advanced systems develop plans by predicting certain rewards and the mere anticipation of reward is sufficient to determine action, which, if all goes according to plan, will eventually be received. C3 and C4 have the right kind of mechanisms to engage in this sort of activity.

fact that rewards strengthen rules. When a system takes a certain action that is rewarded, the rules causing that action are strengthened, making it more likely that the same rules will successfully compete to post their messages in successive encounters with the same conditions. Eventually the reward is back-propagated by the bucket brigade through the entire sequence of rules leading to the action, meaning that the system will be more likely to act so as to obtain the reward in the future. The larger a reward is, the stronger the rules leading to that reward become and the more numerous such rules become because of the preference given to strong rules by the genetic algorithm when it chooses its parent population. The strongest rules determine actions leading to the greatest reward, and the actions with the largest number of rules involved in activating them also lead to the largest rewards, precisely because the size of the reward determines the strength of the rules and the number of rules leading to the action resulting in that reward. Thus the larger a reward is, the more likely it is that the system will act so as to obtain that reward in the future. In other words, systems are disposed to act³⁷ to obtain the greatest reward available to them in the conditions they find themselves.

2.4: Concluding Remarks About C1

C1's actions are based on information it has about its world, but it has no representations of that world. In this sense it is merely an automaton; however, unlike most automata, C1 has a procedure for altering the strength of its rules, so it can change its actions based on reward it receives from the environment through the messages in its detectors. In other words, C1 can exploit its brain plasticity in order to learn, though it learns complicated sequences very slowly. We turn now to developing new systems by adding mechanisms to C1 that can make better use of environmental reward to modify their cognitive architectures. The resulting systems have cognitive features that are more human-like.

³⁷ The reason I say they are disposed to act to obtain the greatest reward available, rather than that they do act to obtain the greatest possible reward, is that the strongest rules do not always win the competition to post their messages.

CHAPTER 3

C2: DECENTRALIZING CONTROL

The first significant development to the structure of C1 takes advantage of the subpopulations formed by the sharing of rewards and payments. By making some small changes and adding one mechanism to the structure of C1, we can develop a new system, C2, that has subsystems within its brain. This system has a less centralized brain than C1 because it has a different message board in each subsystem. As a result, C2 is largely parallel, so it is able to process much more information, i.e. more classifiers, in a given time than C1. With the emergence of subsystems, C2 develops classifiers that send messages between subsystems by which it coordinates their actions. These messages are generated from within each subsystem, so we get integrated action without a command centre.

3.1 Restricted Mating³⁸

Forcing classifiers to share their rewards gives C1 a stable subpopulation, as we saw. A number of rules form that are able to receive a certain reward, the number in proportion to the size of the reward. Many other rules co-adapt to these strong rules by becoming members of sequences that activate them. In this way they indirectly share the reward and become strong themselves. This creates a subpopulation that is fairly independent of the other activities going on in the system. Under the appropriate conditions, members of the subpopulation are activated, and eventually produce an action that is rewarded, or at least one that was rewarded in the past. If the action is still beneficial to the system, the system generates new, possibly better, variations of the rules that lead to that action via its genetic algorithm, by choosing members of that subpopulation to reproduce. It might turn out, however, that members of different subpopulations are actually paired for crossover, since pairing is random. In this case, two strong rules can produce very weak rules. We can see an example of unsuccessful mating by reconsidering our GA with evaluator function $E(x)$. In the early generations, the GA could produce the

³⁸ This mechanism is presented in Goldberg 1989, pp.192-7.

relatively fit strings 0100000000 with fitness 19.7136 and 1111111111 with fitness 10.4329. If these strings mated with a crossover site 2 and no mutation, the result would be the two very weak strings 0111111111 with fitness 3.5721 and 1100000000 with fitness 0.4624. In this case, the parent strings correspond to points on opposite sides of the minimum of the quadratic function in $E(x)$. When they cross over, components of each parent string are passed to the children, resulting in strings very near the minimum. The production of such very weak strings can be avoided by restricting mating to strings on the same side of the minimum, i.e., to certain subpopulations of strings.

Successful creatures often evolve by using existing mechanisms in some novel way, and C2 is designed from C1 with this in mind. C1 has a procedure for comparing strings and determining whether or not they match. In C1 this mechanism is used exclusively for determining which classifiers are activated by the messages on the message board. C2 extends the use of this mechanism to restrict the possible mates a classifier can have. Two classifiers are allowed to mate only when their condition parts match each other. That is, classifiers can only mate with each other if in each position of their condition parts, they both are 0, they both are 1, or at least one of them is #. So the classifiers 110##:10011 and #1001:00110 could mate with each other; whereas, the classifiers 10###:01011 and #110#:00111 could not mate because the first has a 0 in the second position of its condition part and the second has a 1 in that position.

It may seem odd that the message part of the classifier is not involved in the ritual of who gets picked for mating, and indeed, some genetic algorithms use two-part strings to restrict mating, in which the matches must be between the first part of one string and the second part of the other. This sort of mechanism would be of little use to a classifier system, however, since it would mean that only rules that activated each other could mate. The effect of restricting mating to classifiers whose condition parts match is that classifiers in one subpopulation tend to mate only with other classifiers in that same subpopulation. The reason for this is that subpopulations consist of co-adapted classifiers, which means that the classifiers develop message parts and condition parts that match each other. That is to say, subpopulations contain rules that are coupled. Two rules whose

condition parts match each other are very likely, though not guaranteed, to be satisfied by the same message, which means that both rules are likely to be coupled to the same classifier, and hence in the same subpopulation.

Since rules that are chosen to reproduce are relatively strong, they must belong to sequences that produce actions leading to reward. That means that the message parts of parent rules either activate the effectors in a useful way themselves, or they activate other strong rules that belong to a sequence leading to useful action by the effectors. Restricting mating to classifiers whose condition parts match will either pair the messages with different conditions in the new classifier, if the crossover point occurs in the condition part, or it will produce different messages that can be posted under the same conditions, if the crossover point occurs in the message part. In the first case, the conditions will be very similar to the ones in which the message is already successful, since the condition parts of the mating parent rules match each other. Pairing existing messages with different conditions in a new classifier is C2's way of testing to see if the message is also useful in the different conditions. Unlike the situation with unrestricted mating in which a strong message might end up being tested for usefulness in any conditions, restricting mating has the effect of testing strong messages much more "intelligently", by only testing a message in conditions similar to the ones in which it is successful. Restricted mating allows C2 to explore just how useful a message is; i.e., in what range of conditions does it activate useful actions on the part of the system? The second case, in which crossover occurs in the message part, allows C2 to search for new messages under given conditions. Since the messages from the parent classifiers are useful in exactly those conditions, the result of recombining those messages by crossover should be new messages that are also useful in those conditions. This follows because the new messages inherit the building blocks that made their parents successful and they belong to classifiers with the same condition part as their parents. Thus, it is very unlikely that C2 will produce weak rules from strong parents, as can happen, we saw, with unrestricted mating.

The development of subpopulations by sharing is a system's way of specializing its resources. Using restricted mating, C2 makes this specialization more efficient by greatly

reducing the chance of very weak rules being produced from strong parents in different subpopulations. Members of different subpopulations can post messages on the message board at the same time, but every rule trying to post its message is in competition with every other activated rule. Yet there seems to be something odd about having two rules that are activated by messages posted from different detectors and contained in sequences that activate different effectors in competition with each other. Certainly subpopulations should not act in isolation from each other and the various detector messages should be used to integrate effector actions, but direct centralized competition of very different specialist subpopulations results in unnecessary competition of messages activated by entirely different conditions. It would be an advantage to a system only to have messages activated by similar conditions compete with each other, allowing very differently activated messages to be posted independently of each other so that they can determine the system's actions together. This would help prevent the system from failing to take action pertinent to many of the conditions it had discriminated. C2 avoids the direct centralized competition of all messages found in C1 by developing subsystems, each with its own message board and competition for rules to post their messages.

3.2 Subsystems: Cobbling Together Basic Classifier Systems

In C1, every rule can potentially activate every other rule. When a message is posted, the condition part of every classifier must be checked to see if it is satisfied, thereby activating the classifier. Physically this means that C1's brain must be totally interconnected. This is very much the "Jack of all trades, master of none" model of cognition. There are advantages to having specialized subsystems in the brain. First, there is the efficiency of having different subsystems operating in parallel, though not in isolation, so that only rules co-adapted to certain detectors and effectors ever compete with each other. The system can better take advantage of its specialists by having them work together, rather than having them diffused amongst the total population of classifiers. Also, much of the interconnectedness of C1 is redundant because its specialist subpopulations operate largely independently of each other in any case. Subsystems allow C2 to process more information more quickly than C1 by not requiring every rule to be

tested for a match with every message posted. But perhaps the best argument of all for introducing subsystems is that humans have subsystems in their brains.

A metaphorical look at how a system could form so that it had subsystems goes as follows. Two or more C1's join together to form a C2. They do this by connecting with each other, detectors to effectors. Some of C1₁'s effectors attach to some of C1₂'s detectors, so that C1₁ sends messages directly to C1₂. When C1₁'s effectors perform some action, C1₂'s detectors sense this and post a message on the message board of C1₂. A similar arrangement of C1₁'s detectors with C1₂'s effectors allows the C1's to send messages back and forth to each other.³⁹ Eventually the genetic algorithms will produce good rules for the C1's to send messages to each other and a C2 is born, with the C1's acting as its subsystems. C2 is a creature with local control centres for specialized actions. It has the ability to pass messages between its various subsystems, but it has no central control. The hard work is done by the subsystems themselves based on the messages they receive from the environment and the other subsystems.

We can make the notion of a subsystem more precise by adding a short identifier code to the condition and message part of each classifier.⁴⁰ The code need only be long enough so that different values can be associated with each message board. The code attached to a condition part of a classifier would serve to indicate which message board should be checked for matching messages; and the code attached to the message part would indicate which message board the message will compete to be posted on. By including wild card characters in the code specification it is possible for a condition or message to belong to more than one subsystem, though every activation of a classifier affects its strength so most classifiers would belong to only one subsystem, both condition

³⁹ Notice that the effectors that join the two C1 systems become ordinary messages in the newly created system. These messages operate in a very special way, however, because they are activated by messages in one subsystem and post their messages in the other subsystem; i.e., their condition parts match messages on one message board and their message parts are posted on the other message board. This allows the system to integrate the actions of its subsystems when determining an action.

⁴⁰ This type of code is very similar to mating tags and templates used in genetic algorithms (Goldberg 1989, p.192-7).

and message part.⁴¹ Thus this subsystem structure is not completely modular, and it could even include a system with a largely interconnected core, one main subsystem, with more specialized subsystems adjoined. Of course, the system will function better if its specialist subsystems do not have to operate in isolation. It can pass messages between subsystems by having classifiers whose condition parts are activated by messages in one subsystem and whose message parts are posted in another subsystem. Since this is the basis of integrated non-centralized action, it is worth further consideration.

3.3 A Closer Look at Subsystem Interaction

The messages passed between subsystems serve to guide actions in a particular subsystem based on what is going on in the other subsystems. These messages alone allow C2 to coordinate actions; they do so quite simply by ensuring that some of the messages active in one subsystem belong to classifiers whose condition part is activated by messages in other subsystems. So what any one subsystem will do depends on what the other subsystems are doing at every stage. By posting and receiving messages back and forth the subsystems cause each other to act based on the states of every subsystem; each subsystem functions in light of feedback from the other subsystems.⁴² It is clear that the messages posted between the subsystems have a pivotal role to play in the overall functioning of C2. They are C2's sole means of performing complex actions that integrate several effectors controlled by different subsystems, and they allow C2 to determine such complex actions by using messages from many detectors.

The influence subsystems have on each other coordinates their activities, resulting in integrated actions by C2 in the following way. Messages between subsystems will be

⁴¹ This specification of subsystems may seem ad hoc but it is no more so than any other part of the operating system, such as the genetic algorithm, or bucket brigade. I am simply employing reverse engineering by positing a mechanism for realizing a structure present in the human brain.

⁴² This interaction between subsystems is very much like a connectionist system, in which the state of the entire system affects every component of the system. As the strengths of rules get revised, some couplings become very strong and others are eliminated by the replacement of weak rules, which is very much like the changing weights between various nodes of a connectionist system. Thus, it seems that there is potential for my model to include a connectionist architecture, which suggests an interesting direction for future investigation.

replaced unless they are strengthened by the bucket brigade, i.e., unless they belong to a sequence leading to an action that is rewarded. Strong rules successfully post their messages more often and so come to determine what actions the system will take in certain conditions. Hence what is occurring in one subsystem partially causes useful actions by another subsystem. However, every subsystem develops good rules for the conditions it encounters by strengthening rules that lead to reward. Thus the same conditions occurring in a subsystem A that result in a message being posted in another subsystem B such that the message is a partial cause of useful action in B also lead to some useful action in A. By strengthening what works, the subsystems learn to take useful actions together.

The process of developing coordinated useful actions between various subsystems under certain conditions is a slow one for C2 because it must learn simultaneously what actions to take in each of its subsystems. It is like trying to hit a moving target. The problem is that whether or not the action a subsystem takes is useful depends on what actions the other subsystems are taking. Now since they are also learning what is useful under the current conditions, they may change their actions during successive encounters with the conditions. This could mean that the useful action learned by one subsystem ceases to be beneficial, forcing it to change as well, which will include altering what messages it posts in other subsystems, thereby affecting their actions also. Thus under certain conditions, i.e. a given set of messages posted by the detectors, the system develops the rules in its subsystems based in part on how the other subsystems are developing. After the learning period, which could be quite long and require many encounters with the initiating conditions, the actions of individual subsystems are integrated to produce a coordinated action by the entire system. Though this learning to integrate the actions of its various subsystems is a slow process in C2, it has the important feature that no central control is required to achieve this integration. Coordinated actions are produced by subsystems developing their rules together, based in part on the messages

they receive from each other.⁴³

3.4 The Role of Schemata in C2

Recall from section 1.1.5 the notion of a schema, which is simply a set of bit strings, all of whose members have the same specified value in certain positions. This notion is easily extended to classifier by specifying whether the relevant pattern of specified position values occurs in the condition part or the message part of the classifier.⁴⁴ Now according to the Fundamental Theorem of Genetic Algorithms, highly fit schemata come to dominate the string population. In the context of classifier systems, where fitness is rule strength, this means that the system's classifiers will contain mostly schemata leading to rewarded actions. These useful schemata cause the rules in which they occur to be strengthened via the bucket brigade, by sharing the reward gained from the action prescribed by the sequence containing those rules. Strong rules are selected to reproduce by the genetic algorithm, thereby increasing the number of rules containing those useful schemata. Rules that do not result in beneficial actions for the system are weakened and eventually replaced, eliminating the schemata contained within them from the rules in the system. Thus, it is useful schemata that guide the system's actions. That is to say, sequences of rules form to enable the system to activate its effectors based on messages posted by its detectors, and it is in virtue of schemata that rules co-adapt to form these sequences. Rules in sequences leading to reward are strengthened so these sequences come to determine the system's actions.

A single classifier can have several schemata within it. Some schemata are contained in its condition part and messages containing the same schemata activate the classifier. Note that a condition does not have to match every position in a message because of wild card characters, so it can have fewer schemata in it than the messages that match it. When a classifier is activated, it will compete to post its message, which also

⁴³ Again the image of a connectionist model suggests itself very strongly.

⁴⁴ I will use "schema" to refer to a set of classifiers and the pattern of specified values defining set membership interchangeably. Context will make it clear in what sense the term is being used.

contains schemata, though not the same ones as in the condition (except by coincidence).

The relationship between the schemata in the condition part and the message part of a single classifier is strictly causal; i.e., the schemata in the condition part merely play a causal role in having the schemata in the message part become active in the system by the posting of that message. None of the schemata in the condition part need be included in the message part, so just as in C1, C2's messages are not representations. They are not about the conditions causing them to be activated since no information need be passed between the condition part and message part of a single classifier. A message in C2 is of the form "Certain conditions obtain, therefore take a certain action". The conditions cause the action, but the action need bear no other relation to those conditions. In other words, useful schemata in the condition of a rule merely cause other useful schemata in the message to become active, with no other relation between them.

Information about the environment, which is transformed into detector messages, can be understood to influence the system in terms of schemata as follows. In a sequence of rules leading to some action, schemata in posted messages activate classifiers whose conditions have some of the same schemata. These classifiers then compete to post their messages, which contain other schemata, and the process continues until the effectors are activated. Now it may happen that some of the rules in a sequence post their messages in subsystems different from the ones in which they were activated. In this case, schemata in one subsystem will cause schemata to be posted in another subsystem and thus influence the actions of that subsystem. Since all of the subsystems of C2 pass messages between themselves, the actions of every subsystem are influenced by what is occurring - i.e., by what schemata are active -- in every other subsystem. But many sequences begin with messages posted by the detectors; hence it follows that the schemata contained in those messages posted by the detectors, i.e., those determined by the environment, affect the entire cognitive activity of C2.

3.5 C2's Cognitive Features

Every message of length l contains 2^l different schemata. Because of combinatorial explosion, it is not possible for a system to have as many rules as it does schemata.

Now because of wild card characters in the condition part of classifiers, it is possible to have rules activated by most of the different schemata occurring in the system. That is to say, different messages containing different schemata can activate the same rule provided it has wild card characters in every position of its condition part where the two messages differ. However, in all such cases the activated classifier posts the same message, even though different conditions activate it. What this means is that C2 has limited capacity to act relative to what it can detect. Many of the schemata contained in the messages posted by the detectors will lead to the same action by C2; so while the detectors may be affected differently by conditions in the environment, C2 must act as though the situations are the same. This will also affect subsystem interaction because some of the different schemata occurring in one subsystem will activate the same rule to post its message in another subsystem. C2 cannot discriminate between many of the schemata active in its rules when determining an action.

Another weakness in C2's cognitive architecture results from the fact that the messages in C2 are not representations, as we saw. It may be the case that a single event or object in the environment is the cause of messages being posted by many different detectors. The schemata in these messages will be active in many different subsystems and the actions of C2's subsystems may get integrated, provided C2 has had sufficient opportunity to learn useful actions in the given conditions. What C2 will not have is any way of recognizing that all of its subsystems are reacting to the same environmental event. In effect, each subsystem has to learn about what is affecting it on its own. The subsystems influence each other because of the same external event but they cannot inform each other about that event. As we will see in the next chapter, the mechanism of pass through characters allows C3 to overcome this problem, while increasing the number of distinct actions that the system can take.

The important feature of C2 is that it has subsystems in its brain, which allows it to specialize its brain activities. A consequence of having subsystems in its brain is that C2 needs a way of sending messages between its various subsystems if it is to perform integrated actions. C2 has the subsystems send the messages themselves, meaning that its

brain is massively parallel, with no central controller and no subsystem through which all messages must pass. This makes C2 an important "evolutionary" bridge between classifiers systems and systems with higher cognitive functions.

CHAPTER 4

C3: ABSTRACT THINKER, QUICK LEARNER

C3 differs from C2 by having two additional mechanisms: pass through characters and predictor rules. Pass through characters make it possible for C3's messages to contain representations. The relationship between conditions and messages within a single classifier is not just causal as in the case of C2. Information can now be passed from a condition to a message. In particular, when different schemata activate the same rule, information capturing the differences between the schemata can now be passed into the message, meaning that a single rule can be enriched to prescribe different actions, unlike what happens in C2. This greatly increases the number of actions determined by the rules present in the system at any one time. Also, the passage of information from condition to message part in a single classifier is a mechanism which makes abstraction possible.

C3 is also a much faster learner than either C2 or C1. The reason is that we add predictor rules to it, so unlike its predecessors, C3 does not need to repeatedly encounter the same conditions in the environment in order to strengthen an entire sequence of rules via backchaining in the bucket brigade. A single encounter is often all C3 needs to learn complicated and useful sequences of rules. Predictor rules operate by simply predicting what a certain detector will post when the system takes some action. Individual predictor rules cannot predict novel actions by the environment, but novel conditions activating novel collections of predictor rules can result in a collective prediction of an environmental event never experienced. C3 is a much more advanced system than C2, and while it does not yet have mechanisms that exhibit the characteristic features of awareness of itself, its mechanisms produce actions from something like awareness of its environment in a manner resembling intelligence.

4.1 C3's Information System

4.1.1 Pass Through Characters⁴⁵

C3 has a greater degree of plasticity than C2 or C1 because the message parts of its classifiers are not fixed for each classifier. Messages in C3 include a pass through character, *. The pass through character is a variable that takes a value 0 or 1, corresponding to the value of the condition part of the classifier in the same position in which the * occurs in the message part. For example, the classifier 01001:1*100 posts the message 11100, because the second position in the condition part of the rule, corresponding to where the * occurs, is a 1. It is clear from this example that there is no advantage to having a pass through character in a position corresponding to a fixed bit in the condition part of the classifier. In every instance the same message will be posted since the value of the corresponding position in the condition part is always the same. The pass through character serves no purpose. The situation is no different than if the bit in the message part was fixed at the value of the corresponding position in the condition part.

Pass through characters are useful, however, when the position in the condition part is a wild card. In this case the bit in the message part takes the value of the bit in the same position of the message that activated the rule. Thus, the value of * depends on what message activated the classifier in which it occurs, and since the wild card # matches both 0 and 1, * can take on either of those values in different conditions. For example, if the classifier 110#1:011*1 is activated by ##010:11001, then it will post the message 01101 because the fourth position in the message part of the activating classifier is 0. However, if the same classifier is activated by 00##1:11011, then it will post the message 01111. Having the classifier 110#1:011*1 is just like having the two distinct classifiers 110#1:01101 and 110#1:01111, except C3 saves memory space by needing only one rule where C2, with its lesser plasticity, needs two; i.e., pass through characters act as variables. They provide C3 with a very compact way of storing many rules. In fact, a classifier in C3 with k positions in its message part being *, can post 2^k distinct messages.

⁴⁵ This mechanism is presented in Holland et al 1987, though the use I make of it in this work is entirely original.

It is possible that many consecutive rules in a sequence have pass through characters in the same position. By forming such a sequence C3 could pass the value of a bit in a message posted by the detectors all the way to the effectors. More importantly, C3 could pass on several bits through a sequence by developing a co-adapted sequence with several pass through characters in the same positions. In particular, it is possible for C3 to form strings that pass on useful schemata, and because one string can pass on several different schemata, C3 does not have the problem C2 has of its useful schemata being able to activate only a small number of distinct classifiers. C3 uses the same small number of classifiers as C2, but it can post 2^k times as many distinct messages by having k positions in the message parts of its classifiers occupied by pass through characters. So not only can C3 post a significantly larger number of messages than C2, it can actually pass on parts of messages directly, which, because of the bucket brigade, will eventually be the most useful or efficacious schemata. Now because C3 also has the structure of subsystems present in C2, it can develop sequences of rules that post messages containing the efficacious schemata in the messages posted in one subsystem on the message boards of other subsystems. Thus C3 has a quite efficient way of sending messages between subsystems indicating the essentials of what is going on in a particular subsystem, while discarding the details relevant only to that subsystem. This ability is the essence of abstraction, but before turning to that notion we must consider the content of the rules in C3.

4.1.2 Semantic Content: The Formal Approach

Virtually all current theories in cognitive science rely on a formal account of content, motivated by considering the mind as a kind of computer. Two important assumptions characterize this approach to cognition:

1. The brain is a machine that operates on syntax;
2. Human cognition operates on two distinct levels, a semantic level and a syntactic one.

Even philosophers with positions as different as Dennett and Fodor do not disagree on these basic assumptions. Regarding the first, Dennett says that "the brain ... is just a

syntactic engine ... That's all brains *can do*" (Dennett 1990, p.61); while according to Fodor "our best available theory of mental processes -- indeed, the *only* available theory of mental processes that isn't *known* to be false -- needs the picture of the mind as a syntax-driven machine" (Fodor 1990, p.23). The important issue surrounding the second assumption is made clear by Garfield.

Symbols have a dual nature. On the one hand, they are physical things (like the ink on this page, or the electrical impulses and magnetic records in the computer); on the other hand, they stand for things other than themselves. The computer processes these symbols according to rules, and the meanings of the symbols are tied up with these rules. But the computer does not need to "know" the meanings of the symbols. It performs its operations on the symbols by means of procedures that depend only their physical characteristics. The trick, of course, is to get the physical and meaningful (or semantic) characteristics of the symbols, the rules, and the machine employing them to match up in the right way. This is the essence of an information-processing system: that it encodes information about the world, operates on that information in some way that can be characterized as meaningful, and is structured as a set of functionally organized, interacting parts. The digital computer is a perfect example of these ideas, and it has provided dramatic evidence that intelligent performance can be the product of a physical information-processing system (Garfield 1991, p.313).

As Garfield points out, the "trick ... is to get the physical and meaningful (or semantic) characteristics of the symbols, the rules, and the machine employing them to match up in the right way". Dennett and Fodor agree that there are two distinct levels involved in cognition and they agree that semantics and syntax have to match up in some way. They even agree that semantic content is not in our heads. What they disagree about is *how* to get semantics and syntax to match up in the right way. This is a central question in cognitive science, one that anyone advancing a model of the sort we are must come to terms with. However, before addressing the question directly, we consider the solutions offered by Dennett and Fodor and the shortcomings of these positions. This will give us some framework from which to offer our own solution.

According to Dennett,

"... the brain, as physiology or plain common sense shows us, is just a *syntactic engine*; all it can do is discriminate its inputs by their structural, temporal, and physical features and let its entirely mechanical activities be governed by these "syntactic" features of its inputs. That's all brains *can do*. Now how does the brain manage to get semantics from syntax? ... It

couldn't. The syntax of a system doesn't determine its semantics... It cannot be designed to do an impossible task, but it could be designed to *approximate* the impossible task, to *mimic* the behaviour of the impossible object (the semantic engine) by capitalizing on close (close enough) fortuitous correspondences between structural regularities -- of the environment and of its own internal states and operations -- and semantic types" (Dennett 1987, p.61).

Dennett's position concerning the second basic assumption is an "as if" behaviourism. If the brain acts *as if* it is a semantic engine, or sufficiently closely "mimics" one, then things match up in the right way. There is no fact of the matter about the semantics of a system, since the content of its symbols can only be ascribed under some interpretation of its actions, and there are always alternative descriptions of a system's actions, no one of which can be favoured in principle. Since all descriptions are equally good, any choice between them is mere legislation. Now under a particular interpretation a system can act as if it "extract[s] semantically reliable results from syntactically driven operations" (ibid). The difficulty with this view is that it presupposes there is something to mimic or approximate, albeit within some interpretation. If so, the difficult questions shift to the interpreter in question and cannot simply be deflected by some "as if" rationale. Within the interpretation something must count as semantics so that a judgment can be made as to whether the actions of the system are "close enough". However weak the notion of "close" Dennett intends to use, some comparative measure is required and that requires having two things to be compared. We have a system and its actions as one of the elements but no physical system can serve as a standard of comparison without begging the question about how it generates semantics from syntax.

In fact, Dennett intends the semantic engine to be a purely theoretical device within an interpretation, i.e. a set of defining characteristics which have no physical realization. After all, people mimic Santa Claus and we can even judge between better or worse enactments of Santa Claus without being committed to the physical existence of Santa Claus. This move will not work for the case of a semantic engine, however; the disanalogy between Santa Claus and a semantic engine is that the defining characteristics of a semantic engine are its contents, the very things we are trying to account for by

appealing to a semantic engine. Unless it is possible to extract semantics from syntax it is not possible for any physical system to be the interpreter Dennett needs. A physical system can have certain states which it transforms according to their syntax and its operating rules. Under some interpretation these states can be about another system such that it interprets the other system to be acting as if its states have semantic content; i.e., under some interpretation there can be the right sort of interpreter. But what of this new interpretation? Unless there is a semantic engine we are, or at least Dennett seems to be, faced with an infinite regress. If no system is a semantic engine then there can be no interpretation from which it acts as if syntactic systems are semantic. But if there is some semantic engine, the questions remain as to how.

Fodor's position concerning the second basic assumption is developed from two premises. First, syntactic structure and semantic properties are isomorphic to each other. Second, there is a causal link between the environment and the symbol tokens in a system. Content supervenes on the isomorphic structure of syntax and semantics because of this causal link. That is to say, content is not in the system, it is added when the system is hooked up to the world. "No symbol means *cat* unless it carries information about cats. But no symbol carries information about cats unless its tokenings are somehow nomically dependent upon cats" (Fodor 1990, p.127). Fodor's account of content is as follows:

You connect the causal properties of a symbol with its semantic properties via its syntax... To a first approximation, we can think of its syntactic structure as an abstract feature of its (geometric or acoustic) *shape*. Because to all intents and purposes, syntax reduces to shape, and because the shape of a symbol is a potential determinant of its causal role, it is fairly easy to see how there could be environments in which the causal role of a symbol correlates with its syntax. It's easy, that is to say, to imagine symbol tokens interacting causally *in virtue of* their syntactic structures...

But, now, we know from formal logic that certain of the semantic relations among symbols can be, as it were, "mimicked" by their syntactic relations... So ... the semantic relation that holds between two symbols when the proposition expressed by the one is implied by the proposition expressed by the other can be mimicked by syntactic relations in virtue of which one of the symbols is derivable from the other. We can therefore build machines which have... the following property: the operations of such a machine consist entirely of transformations of symbols; in the course of performing these operations, the machine is sensitive solely to the

syntactic properties of the symbols; and the operations that the machine performs on the symbols are entirely confined to alterations of their shapes. Yet the machine is so devised that it will transform one symbol into another if and only if the symbols so transformed stand in certain *semantic* relations; e.g., the relation that the premises bear to the conclusion in a valid argument. Such machines -- computers, of course -- just *are* environments in which the causal role of a symbol token is made to parallel the inferential role of the proposition that it expresses. (Fodor 1990, pp. 22-3)

Fodor avoids objections to his account along the lines that causation is not sufficient to provide content because a token can be variously caused by introducing the notion of asymmetrical dependence. He claims " that "X" means X if:

1. 'Xs cause "X"s' is a law.
2. Some "X"s are actually caused by Xs.
3. For all Y not=X, if Ys qua Ys actually cause "X"s, then Ys *causing* "X"s is asymmetrically dependent on Xs *causing* "X"s. (Fodor 1990, p.121)

The notion of asymmetrical dependence is that though it is possible for both Xs and Ys to cause "X" tokens, they are "X" tokens in virtue of being caused by X in some earlier situation. In other words, "X" can only mean X if it is caused by X at least some of the time. "X"s take their content exclusively from being caused by Xs and if causal link between "X"s and Xs were broken then the token would not be "X". Put another way, a system could not form a representation of X independent of X, so the token "X" derives its content from being caused by Xs even though it could also be caused by Ys. Faced exclusively with Ys, a system could not forge the "X" -> X connection.

Fodor's account of content may be sufficient to do much of the job of connecting things up in the right way so that content can arise from the purely syntactical operations of a system. It may be more than sufficient, however; i.e., the underlying assumptions may be too strong. Fodor's account is lacking in two important areas, and while these are only practical concerns in the sense that nothing in principle precludes the possibility of extending Fodor's account to address them, they make the account considerably less compelling, especially in light of an alternative. First, the causal link between the environment and a system's symbol tokens, including asymmetrical dependence, is far more plausible when the Xs and Ys are cats and dogs or cows and horses, than when they are more abstract objects like Hilbert spaces or anti-matter. Fodor admits that it may be

virtually impossible to determine the causal relations involved in fixing content (Fodor 1983), but all that matters is that there is such a relation. In the case of objects we can perceive, the existence of such a relation is something we can accept intuitively. It seems entirely possible that some causal relation might be involved in determining the content of a symbol token "dog" even if the nature of that relation is unknown, simply because dogs are the sorts of things that can cause other things in a very straight-forward way. It is far less plausible that a causal relation is involved in determining the content of a symbol token of some abstract object. In what way does a Hilbert space cause a token symbol to have the semantic content "Hilbert space"? Granted that this is not a question Fodor needs to answer, but he does need to provide *some* indication that this causation occurs, for without that he is asking us to accept it on faith. But his is not the only game in town, so the burden of proof rests with him to make plausible the notion that abstract objects cause token symbols to have the appropriate semantic content; otherwise, much of semantics is left unaccounted for.

A more serious deficiency in Fodor's account of content is that he offers no explanation for the isomorphism between semantic and syntactic structure, yet everything rests on this assumption. For example, there can be no asymmetrical dependence without this isomorphism because the causal relation between the environment and a symbol would be purely syntactical. Now in computers, "the causal role of a symbol token is *made* to parallel the inferential role of the proposition that it expresses" (Fodor 1990, p.23, my emphasis), but we make them parallel. How is it that nature performed this miracle in us? It is clear that any creature whose semantics and syntax happened to parallel each other exactly would have a huge survival advantage over other creatures. If semantic and syntactic structure were quite simple it is possible that they could have come to parallel each other exactly by chance, after which this fortunate coincidence could have been chosen for by natural selection. But the semantic and syntactic relations of symbols in the human mind form the richest structures we know of. It is not an accident that they parallel each other exactly and an important part of the story of content is missing if this isomorphism is not accounted for. Fodor just posits this isomorphism because he needs it, yet explain-

ing it is an essential component of any theory of content. Fodor's silence about this central issue raises questions not only about how the isomorphism is achieved but whether or not Fodor has, in fact, presented a correct account of content.

Underlying the formal approach to content is the notion that content is not in the system; the system is a purely syntactic machine. A relation to something external is needed to provide semantic content -- for Dennett the relation is to an interpreter, for Fodor it is to a world; however, since syntax is all that a system can manipulate, syntactic structure must parallel semantic structure. The difficulty in developing a theory of content is to account for this isomorphism. Our model will sidestep this issue by taking the commonsense realist position that there is content in the system. We take this position not to be philosophically controversial, though that is undoubtedly a consequence, but because the mechanisms in C3 suggest it. Rather than having to account for an isomorphism between semantic and syntactic structure, we show that when there is content in the system the distinction between syntax and semantics collapses. C3 manipulates strings of information so that syntactic and semantic structure are not isomorphic, they are the same. To see how information is active in C3 we must consider how C3 forms representations and how it performs abstraction.

4.1.3 Abstraction: A First Look at Forming Conceptual Categories

As we saw in section 2.2.2, the messages posted by the detectors contain information about the environment. These messages activate other messages, thereby causally influencing the actions that the system takes. In C1 and C2 the only influence that information about the environment can have is this type of causation, since the other messages in the system are determined independently of this information and so do not contain it. In C3, on the other hand, the value of bit positions in a message can be determined directly by information about the environment because of its pass through characters. Now since schemata appearing in a message posted by a detector correspond to a portion of the information contained in that message in that they are the system's encoding for that portion of information, C3 can pass that information from one message to another by passing schemata. Passing schemata through sequences of rules is the mechanism by

which C3 performs abstraction.

Abstraction is the ability of a system to form general categories by which it can classify particulars. Our introspective experience of abstraction makes it seem like a synthesizing process in which many "unconnected" pieces of information are put together in some intelligent way. We seem to be able to form categories of objects by making judgments about their essential characteristics. The essence of objects in a category seems to be something we add to the particular things that belong to the category. It seems to be one of our higher cognitive activities. However, C3's pass through characters allow it to form categories like those resulting from human abstraction purely by extraction. Rather than being a process of synthesizing information, abstraction in C3 is a process of selectively removing information. C3's abstract categories are not superadded to the information about particular things, they are contained within that information, as schemata.

So how is it that abstraction works? Every message of length l contains 2^l different schemata, each of which could potentially correspond to a category in C3. That is to say, the information corresponding to a schema could determine category membership. Clearly something belongs to such a category if information about it is transformed into messages containing the appropriate schema. In other words, a message containing some schema contains information of a particular instance of something belonging to the category corresponding to that schema. The schemata in C3 that actually determine categories are the ones that are efficacious. Some schemata lead to more useful actions by the system than others. Those that do lead to useful actions become efficacious because rules co-adapt to these schemata, allowing them to be passed through and eventually determine some action. Thus, efficacious schemata are extracted from the original messages in which they are posted and passed into other messages. In this way some of the information in a message is passed on while some is left behind; i.e., some information can be active in the system and some cannot. The information that can be active

determines the system's categories.⁴⁶ Additional meaning of a category is determined by the relation of the information characterizing it to all other information in the system. This relation is just the syntactic relations of the schema corresponding to the information determining the category to all other schemata in the system, since that schema is the system's encoding for the information. Now since rules are often replaced by new rules, the meaning of a category is not static, it changes as new rules emerge and form different relations between schemata by the way in which they interact.

Abstraction in C3 relies on different subsystems using the same encodings. That is to say, information must be transformed into messages containing the same schemata in different subsystems if it is information about the same thing; otherwise, the system could not identify the different schemata as information about the same thing. For example, any message in any subsystem that contains content of a dog must include a schema corresponding to "dog", which must be the same in every subsystem. This means that whether a system hears barking or sees Fido, the messages in the auditory and visual detector subsystems containing that information must include the same schema "dog". "Dog" is abstracted by extracting a schema posted in any message containing information about a dog.

Why should it be the case that different subsystems, possibly very different, post messages with the same schemata in them? The best way to answer that question is to turn it around. How could any system possibly have messages from different subsystems leading to the activation of the same effectors if the messages did not have some patterns in common? They couldn't. If two messages posted by different subsystems activate a certain effector in the same way, then they must each satisfy the condition part of the rule activating the effectors. But this means that they must have a pattern in common; in order to match the specified bits in the condition part they must have the same values in the positions of those specified bits. The common pattern is a schema that the messages share. Now since schemata corresponding to categories are efficacious, it follows that different

⁴⁶ This notion will be made more precise in the next section.

subsystems must use the same schema to encode the same information. This is particularly important for allowing the subsystems to operate in coordination with each other. They do so by passing messages back and forth. Included in those messages are efficacious schemata and because subsystems encode information in the same way, regardless of where information originates it is treated in the same way by any given subsystem. That is to say, the information is encoded as a particular schema and it is passed from one subsystem to another via that schema, so information is always treated in the same way by any subsystem. If different subsystems did not encode information by using the same schema, then what a subsystem did with the information would depend on the origin of that information since different schema are related to the rules within a subsystem differently on the basis of their syntax. The important point is that subsystems can work together by passing information between one another using the same schema to encode the information because individual instances of information can cooperatively influence the actions of the subsystem in virtue of the common schema.

It should not be surprising that different subsystems post messages containing the same schemata, since they should have evolved to work together in the systems that have them. The original messages posted by the detectors are not parts of rules that can be learned. These initial messages are given to systems by their detectors, and systems whose detectors did not post messages containing the same schema⁴⁷ to encode the same information would be less likely to survive natural selection than those that did. Detectors that post messages useful to the system to which they belong are more likely to survive, through their system, than other kinds of detectors; and messages are more useful if large numbers of them from many different detectors can be integrated by a system to serve as a basis for its actions. To do this the information in the messages must be correlated,

⁴⁷ This is an oversimplification of the situation. In fact, it is unlikely that very specialized receptors in highly evolved detectors, such as the human eye, all post messages containing the same schemata when detecting a certain object. It is sufficient, however, that some schema corresponding to the object is produced within the detector subsystem, and that all subsystems use the same schema to correspond to that object. This could be achieved by having many specialized receptors activate the *same* rule containing the schema corresponding to the object. Clearly the notion needs developing but, at least it suggests that the oversimplification is not intractable.

which means that the messages must contain some of the same schemata. Natural selection favours systems whose detectors encode the same information by using the same schemata. This process is not the miracle it might seem. Any message posted by a detector is totally specified, i.e. it has no wild card characters. There is only one message that can be different from any given message in each position. Thus all of the other messages, $2^{\ell}-2$ for messages of length ℓ , must agree with the given message in at least one position. Natural selection favours systems whose detectors exploit this commonality when encoding information, so that after a large number of generations highly evolved systems will have extremely high correlations of the schemata used to encode information in all subsystems.

In order for a schema to be passed through an entire sequence of rules, every classifier in the sequence must have wild card characters in the positions of its condition part that correspond to the specified bit positions in the schema and pass through characters in those positions of its message part. Since rules are generated by the partially random processes of a genetic algorithm, it is more likely that rules will form allowing shorter schemata to be passed through sequences of classifiers rather than longer ones. This means that schemata actually passed through a sequence will usually contain only a small amount of the information in the original message in the sequence. Thus the categories a system develops will be quite general. By this simple mechanism of ignoring considerable amounts of information, extracting only the most efficacious schemata, C3 is able to form categories of the same ilk as those characteristic of human abstraction.⁴⁸

4.1.4 C3's Representations; and Abstraction Again

⁴⁸ An objection to this account of abstraction is that it cannot handle the most interesting cases: it may work for explaining how we arrive at *descriptions* of things but fails to capture the "pictures" in our mind's eye. A two-dimensional array on the surface of our eyes is transformed into a three-dimensional representation in our mind. This process seems to require adding something rather than extracting it. As Brook succinctly put the problem, "It is like the difference between a paragraph and a holograph". My short answer is simply to be a reductionist about phenomenology. No matter how it seems, there is no three-dimensional picture in our heads and all that we require is a representation with the content pertinent to three-dimensionality to account for our experiences. In any case, this problem faces all of cognitive science and is not a feature peculiar to my model so I will leave it open-ended for now on the hopes that any good solution will be one I can incorporate into my model.

In addition to making it possible for C3 to form categories, pass through characters can transform its messages into representations, which none of our systems has so far. Some schemata can be passed from one message to another, which is the system's way of processing information because a given schema is the system-wide encoding for certain information. A message that has a schema passed into it is not completely determined by that schema, however. In fact, there would be no point in passing the schema along if it were; the original message would suffice. Passing a schema into different messages changes the context of the information by embedding it in other information. Now since the other information is not received from the detectors it cannot be information from the environment and must be information from the system. Much of this information will be very specific, activating appropriate rules because of the information passed through from the environment, but being specific, its role is purely causal and it is not passed on to other messages.⁴⁹ Some of the information can be of a more general nature, however. In particular, a message can include a tag indicating the subsystem in which the information originated. Of course, the tag must be adjoined to the information while it is in that subsystem or the information about its source will be lost. A tag is itself a schema that simply distinguishes the original subsystem posting the message. That is to say, the presence of a tag in a message functions within the system to inform C3 of the source of the remaining information in the message. Since every subsystem can have a unique code associated with it, a tag could simply include the appropriate code. However, each subsystem effectively detects the other subsystems as external objects, in fact, part of the object C3, so the tag would actually include information like "C3's detector D₁". Information about C3 is included in the tag in the same way that "the dog's ear" contains information about the dog. So we have that a single message can contain information about something in the environment, a tag indicating where the information originated and information about what the system should do because of this information.

⁴⁹ Such information has an extremely important role in determining what actions the system will take but since it is concerned with subtle distinctions between broadly determined actions we will not consider it further. Getting a coherent account of general actions is sufficiently ambitious for this project.

Messages that contain tags have all of the components necessary for them to be representations. Recall from section 2.2.1 that a representation has three components: reference, sense, and function. Clearly a message containing some extracted schema *S* contains information about something, namely the information that the system encodes by using *S*. Now a tag actually allows a message to refer to what the information in *S* is about simply by identifying the subsystem in which *S* originated because this indicates the source of the information, which is the thing in question. For example, if *S* is the encoding of the information "large brown dog", then a tag indicating that this information was posted by the visual detector, contains the information "posted by the visual detector", so that the message contains the information "large brown dog detected by the visual detector" which is the very thing the information in *S* is about. Now the information in any message is information for all of the classifiers that message activates, so the sense and reference contained in a message are sense and reference *for* something. Thus by definition, messages with tags contain representations.

Because of its representations C3 can discriminate between an external and an internal world. Quite simply, if a tag indicates a detector subsystem as the source of certain information than that information originates outside of C3, otherwise it originates inside. This is possible because the information "out there" is equivalent to the information "detected" to C3. Now every message that contains a tag indicating some detector contains the information "detected" because the tag is the encoding for the information "detected at detector D"; so every message containing a tag indicating a detector contains the information "out there". All other information originates "internally", so if a non-detector subsystem posts a message containing information of a large brown dog that the system includes in a representation, it can discriminate this "mental construct" from a dog in the world and act accordingly.

We now return to the notion of abstraction. Based on the structure of a representation, we can make the notion of a category more precise. Recall that abstraction is the process of identifying similarities between things by extracting schemata common to the encoded information about the things. In fact, C3 does not explicitly identify things as

belonging to a certain category in the way that we do, it simply treats them all the same way in virtue of their shared schema, which determines the category. But there are many more efficacious schemata than there are categories; being efficacious is not sufficient to determine a category. Categories are determined only by those schemata that are the sense of some representation. This may seem like an ad hoc restriction, and in fact in the context of C3, it is not important whether this restriction is made. However, in the next chapter we will consider a system with something like abstract concepts resulting from a kind of awareness of its categories. The mechanism is such that the system can only be "aware" of the categories we have identified here.

4.1.5 Content in C3

C3 extracts information from the environment through its detectors. Information is encoded into bit string messages in a lawful way so that the same information is always transformed into the same schema or schemata. A given schema does not just correspond to certain information because of some correlation of semantic properties and syntactic structure. The schema is an encoding for that information. There is genuine meaning in C3's brain. C3 manipulates content by processing schemata. The semantic structure of C3's information corresponds exactly to the syntactic structure of its corresponding symbols because they are both the same structure in C3, namely the relation of schemata to all of the rules in the system. In other words, here syntactic rules *are* inferential. By extracting information directly from the environment C3 is not faced with the burden of having to correlate two incredibly rich structures. It can form abstract categories and represent things in an entirely mechanistic way because it processes information directly.

Because C3 takes content directly from the world, its mechanisms of abstraction and representation give it something very like what Brook refers to as simple awareness (Brook 1994, pp.64-5). Simple awareness is just a system's awareness of the things it encounters. It is important that this is a cognitive response to the environment. Sunflowers move to face the sun but they do not thereby possess simple awareness of the sun. However, when C3 encounters something in the environment, it transforms information about that thing -- in the form of reflected light, heat radiation, sound waves, etc. -- into

messages. Some of the information about the thing is extracted from these messages and passed into other messages. When the information's source is indicated by a tag, the system has a representation of the thing and C3 determines its actions based on that representation and other information about the thing. In other words, the system detects something in its environment and acts directly because of a representation of the thing. This may not be the whole story about awareness but it is certainly the right kind of story. We will develop it further in the next chapter.

4.2 Learning in C3

4.2.1 Predictor Rules

Even with its ability to abstract, C3 has not improved its capacity to learn. For that it also needs a model of its model, which we will develop in C3 by adding what we will refer to as predictor rules.⁵⁰ Predictor rules provide C3 with a way to model its rules and so learn much more quickly than C1 or C2; they are C3's prediction as to what will happen under a certain set of conditions, hence the name. As we have discussed above, the only information C3 has about the environment is contained in the messages posted by its detectors. So if C3 wants to predict how the environment will respond when the system takes some action, all it needs to do is to predict what messages will be posted by the detectors after the effectors are activated. It does this by developing predictor rules that are activated by the same conditions activating certain effectors, i.e., the condition part of a predictor rule matches the same messages that the condition part of a rule whose message activates some effector matches. The message part of a predictor rule is C3's best guess about what results its actions will have. The message includes a tag referring to some detector and the rest of the message is what C3 expects to appear on the message board of that detector, based on its past experience and what is going on in the rest of the system. Of course, this entails the co-adaption of all of the system's rules throughout its

⁵⁰ I have developed this notion of a predictor rule from what Holland refers to as an abstract rule (Holland et al p.75), for the purposes of improving the system's capacity to learn. Holland's notion of an abstract rule is far too intentional for our purposes, including notions of goals and procedures. In fact predictor rules are intended to be a mechanism for what Holland et al refer to as a "lookahead" mode (ibid p. 79).

history, so a considerable amount is involved in any one predictor rule posting its message. Predictor rules never work in isolation, but in order to understand what they are I will begin by considering a single one. In fact, this is not overly misleading because the effects of the rest of the system are implicitly contained within the conditions that activate a predictor rule. It is important to realize that predictor rules are the mechanistic underpinnings of what we call prediction, and that they are very localized. Only when many operate together does something like what we consider to be a prediction emerge. Predictor rules associate the actions of certain effectors with some change in the environment by forging a link between an effector and a detector. Under certain conditions an effector will take some action and under those same conditions a predictor rule will indicate a message should appear in a certain detector, i.e., the environment will cause the message to be posted in the detector because of the effector action.

To see exactly how a single predictor rule operates, imagine a very simple system, one whose detectors post messages one bit long. A specific predictor rule simply indicates which value, 0 or 1, a particular detector will post under given conditions. The crucial point is that under certain conditions the environment always causes the same message to be posted. That is to say, whenever the system activates a certain effector, the environment always responds in the same way in so far as it affects a certain detector in the same way. Now, in our very simple example, it is not surprising that the environment, for all its novelty, is so predictable. Any given detector can only discriminate two states. However, the system can be far more discriminating about when to activate its effector, because the effector is only activated by certain messages being posted and whether or not they will be posted depends on everything else that is occurring in the system. But this requires that several conditions in the system all obtain simultaneously, which will only be the case when certain patterns of detector messages occur, i.e. under somewhat similar environmental conditions. So an effector, and hence a predictor rule, will be activated only in relatively similar environmental conditions. Now since a detector that is of any use to a system, which C3's are because they are the products of considerable evolution, represents similar environmental conditions similarly, the conditions that activate the

effector will be sufficiently similar so that the detector always posts the same message, 0 or 1, when the effector is activated. With only a few trials a predictor rule can get right almost every time what the detector will post in those conditions.

If we consider our simple example above from a combinatorial point of view, we see that a system with m detectors can discriminate 2^m different environmental states, whereas a single detector can only discriminate 2 states. Only some of the 2^m different states will result in messages that satisfy the condition part of an effector. The effector action will cause some change in the environment that the system will detect as one of the 2^m possible message configurations in its detectors, but a single detector can only detect this new state in the environment in one of two ways. The point is that when the effector action is taken, the detector always detects the environment in the same way because the conditions in which the effector is activated are sufficiently similar that a single detector cannot discriminate between the new states produced in the environment by the action. Of course, it need not be the case that the correlation is exact when the system is in its early stages, but in this case the detector will discriminate between the effects of the action and the action will receive less reward in certain circumstances, say when the detector posts a 0. This will have the effect of making the rule controlling the effector action relatively weak and eventually it will be replaced by a rule that only activates the effector in conditions that the action results in a 1 being posted. Once this correlation is established the system can produce a predictor rule indicating that the detector will post a 1 whenever the conditions activating the effector obtain. The point is simply that the system takes actions so as to obtain the greatest possible reward, but since reward comes to the system from messages posted in its detectors, the system co-adapts its rules so that effectors are only activated in conditions in which the message resulting in greatest reward will be posted, *the same message* in all instances.

The only correlation between detector and effector required for the above argument is that the message posted by the detector can influence the strength of the rule activating the effector. In a very simple system, it might be the case that everything is totally interconnected so that every detector is related to every effector in this way. However, as

systems become more complex it may well be that reward is apportioned locally within subsystems, so that some rules could be rewarded while others had their strengths decreased at the same time. Local environmental reward would mean that a single effector action would not be correlated to every detector in the system, but only to a few or even one. Thus single predictor rules would make local predictions. They would not do so in isolation, however, because the conditions activating any effector only obtain as a result of the actions within the entire system, including messages being sent between systems. So the system would effectively settle on a large number of conditions simultaneously from cooperative interaction between its subsystems, leading to a large number of effector actions predicted locally by individual predictor rules operating in the context of the entire active architecture of the system.

When a more realistic system is considered, rather than the very simple one above, the combinatorial argument becomes even stronger. In a system with messages of length l bits, any detector can post 2^l different messages about the environment but the system can discriminate $(2^m)^l$ different environmental states. So for every bit position message length is increased, a single detector increases its discriminatory capability by 2, while the system's capacity is increased by a factor of 2^m . Thus as a system becomes more complex it has increasingly greater power to specialize the conditions under which an effector is activated guided by its disposition to obtain the greatest possible reward, which comes from certain messages being posted by some of its detectors. Thus it is increasingly easy for the system to co-adapt its rules so that the conditions under which an effector is activated lead to maximum possible reward, which implies that the detectors associated with the effector, in that they influence the strength of the rules activating it, post the same messages under those conditions. So even in complex systems, a predictor rule can be developed to indicate what message an effector action will cause the environment to post on a certain detector. C3 exploits its finite discriminatory capabilities in order to make predictions.

So far we have only considered the actions of a single predictor rule. On their own, predictor rules cannot predict any kind of novel action by the environment. A

predictor rule simply mimics what the environment does in certain conditions. However, a single effector action can be involved in numerous different actions on the part of the entire system, depending on the circumstances it is facing. What the environment does will be different in each case. Nonetheless, it is still possible for C3 to predict novel actions by the environment by having many predictor rules active simultaneously. Despite the fact that various combinations of effector movements will result in different overt actions by C3, which could possibly result in some novel action by the environment, at the local level at which a predictor rule operates, the environment will act the same whenever the conditions activating that predictor rule obtain, as we have just shown. What the environment does that is novel is to cause a collection of messages to be posted by the detectors that has never been posted before. Each individual predictor rule makes predictions about what it "knows" from past experience and the overall effect is that the system predicts a collection of messages it has never experienced, i.e. a novel action on the part of the environment. The total set of conditions leading to the individual predictions by the predictor rules serves to provide the theoretical context in which a prediction of the environment's actions can be made.

Even when the correlations between effectors acting and detectors posting messages are well established, C3 will not always get its predictions right. In fact, some of its predictions will be very poor, especially when C3 is first learning sequences for actions and can produce little better than random behaviour. If the predictor rules are to be of any use to C3, it needs a way to change its predictions when they are bad, i.e., it must be able to learn how to make better predictions. C3 keeps a record of the usefulness of a predictor rule in the context defined by the conditions activating it, as it does with any other rule, in its strength value. Strong predictor rules are those that have made good predictions in the past; i.e., the message they predict will be posted is very close to what is actually posted. However, unlike other classifiers, a predictor rule's strength is modified by explicit prediction-based revision, rather than environmental reward or the bucket brigade. That is to say, the message a predictor rule posts is compared bit by bit to messages posted by the detector it indicates. The rule is then evaluated based on its

closest match, where a very close match strengthens the abstract rule and a poor match causes it to be weakened, the amount of strengthening or weakening is proportional to the number of matching bits. The reason that a predictor rule is evaluated by explicit prediction-based evaluation is that a system should be able to predict harmful consequences as a result of certain actions. The other methods of apportioning credit that we have seen would weaken any rule associated with negative rewards, thereby precluding the possibility of predicting harmful results from an action. By this direct evaluation weak predictor rules are quickly replaced by the offspring of strong rules bred under the constraints of restricted mating. In this way, new predictor rules will be activated by conditions activating some effector and they will contain schemata in their message parts that have appeared under those conditions in the past in the detector they tag. Because a predictor rule is strengthened or weakened relative to how good a prediction it makes, the weights are set up so that poor rules are weakened considerably in a single posting, meaning that just a single poor performance by a predictor rule can result in its replacement.

4.2.2 Using Predictor Rules for Single-Instance Learning

C3 is able to use its predictor rules to improve its efficiency in learning. Predictor rules constitute C3's model of the environment; but, so far we have not put that model to any use, we have simply considered how C3 makes predictions and described a mechanism for testing how good a model it has. What C3 can do with its predictor rules is test its other rules, particularly when they result in negatively rewarded actions, without having to encounter the environmental conditions that activate them. The predictor rule can provide feedback in lieu of the environment. As Popper put it, "our hypotheses can do the dying for us". After some poor performance by the system, C3 can "reconsider" the situation by posting messages that will activate the same sequences of rules involved in determining the poor actions. These rules will then compete to post their messages just as in the case when the sequences are activated by messages from the detectors and they will eventually determine some action. Now, since the initial messages are not posted by the detectors, the tag passed through the sequence will indicate this and thereby prevent

2 of/de 2

PM-1 3 1/4" x 4" PHOTOGRAPHIC MICROCOPY TARGET
NBS 1010a ANSI/ISO #2 EQUIVALENT

PRECISIONSM RESOLUTION TARGETS

the effectors from being activated, since the condition for the effector will not be satisfied. This means that sequences determine some action without actually taking it,⁵¹ hence no environmental reward is received. However, the message of the predictor rule contains the content of the message the environment would cause to be posted if the action were taken. By treating the predictor rule message as a message caused by the environment under these conditions, the system can strengthen or weaken the rules leading to that action according to what the environment would do if the action were actually taken. The crucial point is that C3 can repeat this process until it finds a sequence that leads to satisfactory results.⁵² C3 can avoid letting relatively weak predictor rules affect other rules too much by only allowing a predictor rule to reward in proportion to its strength. Consequently, rules will not prematurely come to dominate the competition to post their messages, and alternative rules will be available when a better predictor rule is discovered. By such acts of reflection, C3 can effectively experience the same conditions many times so that rewards can be back-propagated via the bucket brigade and useful sequences of rules for those conditions can be formed. Using its predictor rules to develop useful sequences of action-determining rules, instead of the environment directly, C3 can learn from a single experience. In its very next encounter with conditions in which it performed poorly, it has the capacity to perform much better. C3 has the evolutionary advantage over its predecessors C1 and C2 that a single instance of certain conditions is all that it requires to learn complicated sequences of rules prescribing useful actions.

It is important to note that C3's learning by modelling the environment is constrained by how good its predictor rules are. Weak rules do not match very closely the messages the environment would cause the detectors to post, and so they are not helpful in predicting the outcomes of various actions. Strong rules, on the other hand, make very

⁵¹ Holland et al use the notion of a "virtual" tag "to suppress overt effector actions" (Holland et al 1987, p.99). I am implicitly using the notion of suppression by having a tag ensure that the conditions leading to effector action are not satisfied.

⁵² If all sequences lead to poor performance, C3 will generate new rules and replace the weakest rules with these new ones.

good predictions and can lead to extremely useful actions. It is vital for C3 to develop strong predictor rules, i.e. good practical science, as quickly as possible. Since explicit prediction-based revision is a direct evaluation of how good a predictor rule is, it can alter a predictor rule's strength very significantly in one evaluation, much more so than environmental reward. But every evaluation of a predictor rule requires some interaction with the environment, so baby C3's will spend much of their time testing predictor rules until they develop strong ones. This makes them more dependent on their parents for their survival in childhood than C2's or C1's, but once strong predictor rules are learned, C3's are far more adaptable.

CHAPTER 5

C4: HERE WE ARE

Conscious states are simply mental states we are conscious of being in. And, in general, our being conscious of something is just a matter of our having a thought of some sort about it. Accordingly, it is natural to identify a mental state's being conscious with one's having a roughly contemporaneous thought that one is in that mental state. When a mental state is conscious, one's awareness of it is, intuitively, immediate in some way. So we can stipulate that the contemporaneous thought one has is not mediated by any inference or perceptual input. We are then in a position to advance a useful, informative explanation of what makes conscious states conscious. Since a mental state is conscious if it is accompanied by a suitable higher-order thought, we can explain a mental state's being conscious by hypothesizing that the mental state itself causes that higher order-thought to occur (Rosenthal 1986, pp.335-6).

As we saw in the last chapter, because C3 can pass information between messages and form representations of what the information is about, it has something like simple awareness. In other words, C3 is "aware" of its environment. But C3 has nothing like awareness of being aware of its environment, nor is it aware of itself as itself, i.e., in any way other than as one of the objects in its environment. These additional forms of awareness are what Brook refers to as empirical self-awareness and apperceptive self-awareness (Brook 1994, p.64). Rosenthal suggests that these forms of awareness, which are characteristic of human consciousness, are higher-order thoughts of other mental states. Furthermore, he suggests that the traditional conception that consciousness gives rise to other mental states needs to be reversed, so that, in fact, the other mental states cause the higher-order thoughts. So the picture is that higher-order thoughts are about other mental states, ones that cause them. Combining these notions with the model we have developed so far, a system could have the characteristic features of consciousness as Rosenthal conceives it⁵³ by having classifiers containing information about the information in other classifiers. Representations of the sort in C3 produce something like simple awareness, but a representation of a representation, by virtue of its content, could provide

⁵³ Consciousness as Rosenthal conceives it includes only what Brook refers to as self-awareness. What second-order rules add to the model is something like self-awareness but we follow Brook's more careful distinction of levels of awareness to include simple awareness in consciousness.

a system with something like simple awareness of its awareness. That is to say, such a system could be in a state like being aware that it is aware of something, which is Brook's notion of empirical self-awareness. *"The second-order thought does not itself have to be conscious in order for its first-order object to be conscious.* You can express a thought without being conscious of it, so you can express a *second-order* thought without being conscious of it -- all you need be conscious of is its object, the first-order thought you report" Dennett 1991, p.307). Now if what the system is aware of happens to be itself then it would be aware of representing itself, which is apperceptive self-awareness, a distinctly human characteristic. We develop C4 from C3 by adding second-order rules, which are simply representations of the representations available to C3, as a mechanistic underpinning for higher-order thoughts.

5.1 A Mechanistic Look at Consciousness

5.1.1 Defining Second-Order Rules

Second-order rules are ordinary classifiers -- non-predictor rules -- that contain a special kind of representation: a representation of another representation. Their content alone distinguishes second-order rules from other classifiers. Second-order rules have message parts that can be posted in many subsystems, coded with mostly wild cards, since this is what makes them useful to the system, as we will see below. They are C4's way of quickly disseminating information active in one subsystem throughout all of the other subsystems. We can understand the nature of a second-order rule by considering how a representation of a representation forms. Recall from section 2.2.1 that in order for something to be a representation of *x* it must be about *x* and so make reference to *x* (reference); it must contain or convey information about *x* (sense); and it must do this for someone or something (function). If a second-order rule is to be a representation of a representation *R*, it must refer to all of the components constituting *R*, including the referent and the subject as well as the content. As with all of its other representations, C4 can achieve this in a quite imprecise way, by simply tagging the subsystem in which *R* occurs. The tag indicates the source of the content of the second-order representation, which is information about *R* -- sense, reference, and subject included -- thereby adding

specifying information about R, so that tag and content together depict a particular instance of R; that is to say, the second-order rule in which they occur refers to R. Of course, this depends on there being information about R in the second-order representation. We now show how this could be the case.

If a second-order rule is to contain information about a representation R, it must get that information from somewhere. The only places this information exists in the system are in the messages containing R, so a message containing R must activate the second-order rule -- this ensures that C4 cannot be "conscious" of something that is not one of its cognitive states. A message containing R, contains information about some x and a tag indicating the subsystem in which the schema encoding x was posted, which together determine the sense and reference of R. Now, since both of these can be passed into the second-order rule from a message containing R simply by wild card and pass through characters, the second-order rule could come to contain information about the sense and reference of R, by having this information passed through its condition part to its message part. How much information was passed through would, of course, determine how much information the second-order rule contained about R -- which is an indication of how good the second-order rule is.

In addition to information about the sense and reference of R, the second-order rule must contain information about who or what R is a representation to. However, R is a representation to all of the rules the message containing it activates and the second-order rule has no access to information about which rules these are because they are activated co-temporally with the second-order rule. By the time the information exists in the operating system for the purposes of apportioning credit, the second-order rule is competing to post its message, making it impossible for the information to be passed into the second-order rule. Thus, all that the second-order rule can do is indicate very generally to which rules R is a representation. It can do this by tagging a common location for all of these rules, thereby indicating that R is a representation to rules in that location. Note that there will always be at least one rule to which R is a representation because the second-order rule itself is activated by the message containing R. Now since

the message containing R can activate rules in any of C4's subsystems, the only common location of these rules is C4 itself. In other words, the most specific information a second-order rule can contain about what R is a representation to is the entire system, C4. So by having the tag and content of R passed through to its message part and including a tag indicating that R is a representation to the entire system, a second-order rule can contain information about R qua representation.

We saw above that information about R together with a tag indicating the subsystem in which it occurs provides a second-order rule with information depicting a specific instance of R, so that it refers to R. Now since we have just shown how a second-order rule could come to have information about the three components in virtue of which R is a representation -- reference, sense, and function -- it follows that the addition of a tag indicating the source of this information would allow the second-order rule to refer to R in addition to containing information about R qua representation (reference and sense). The only other condition that the second-order needs to satisfy in order to be a representation is that it has sense and reference for someone or something. But the information in any message is information to the rules it activates, so a second-order rule has sense and reference merely by activating other rules. Thus, it follows from the definition of a representation that a second-order rule is a representation. In particular, it is a representation of a representation.

5.1.2 The Role of Second-Order Rules in C4

Any message containing a representation R, about some x, contains specific information about x by definition. The system uses this information to determine what action it will take. A second-order rule contains information of a special kind. It contains the information that x is being represented in a particular subsystem of C4. This information is of no use to the subsystem in which the representation is occurring since it already has "simple awareness" of this information in virtue of R. That is to say, R contains information about x, which influences the actions of the subsystem. A second-order rule representing R contains the information that information about x is active in the subsystem. But the subsystem is already acting because of the information about x that is

active in it, so the additional information adds nothing that would alter the actions of the subsystem. This means that the second-order rule would not be involved in determining rewarded actions, hence it is of no use to the system when posted in the subsystem S in which it is activated. However, the information is very useful when it is posted in other subsystems in C4 because it informs them about what is going on in S, the subsystem representing x. Unlike in S, the information that information about x is active in S can alter the actions of the other subsystems because it is not just information about x, but about S also. It was not useful to S because S already had "simple awareness" of this information, but the other subsystems do not. Having information about what is going on in S allows the subsystems to coordinate their actions with S, in determining what action to take. Thus, in its capacity as a message between subsystems, the second-order rule can be involved in determining useful actions for the system. Since rules will only survive in C4 if they are useful to it, C4 will come to have second-order rules that are activated in one subsystem and post their messages in other subsystems. Furthermore, since the information about what is going on in S can be useful in any of the other subsystems, second-order rules will develop so that their messages are posted in many subsystems -- i.e., they have codes consisting of mostly wild cards associated with their message parts - - allowing C4 to disseminate information quickly by the activation of a single second-order rule.

The usefulness of second-order rules to C4 is determined by their content. Like C4, C3 can pass information between its various subsystems, thereby coordinating their actions. What C3 cannot do is send useful information about what is going on in one subsystem to other subsystems by a single message posted in many subsystems, like a second-order rule. The reason C3 cannot do this is that it cannot produce a message with generally useful information, i.e., useful to many subsystems. The contrast is made clear with a simple example. If a dog bites C3 it can represent the information "Dog biting C3" in its visual detector. If that information is posted in another subsystem, it must also include some kind of prescriptive information in order to be useful. For example, it could be posted in an effector subsystem that allows C3 to move in a message including some

information to initiate running. Thus, the message posted in the effector subsystem would be something like "Dog biting C3, run". Without the prescriptive content added to the message, the information is useless to C3 because it cannot distinguish a context for it and so cannot determine an action from it. The information "Dog biting C3" could originate in a non-detector subsystem, which would arise if C3 was thinking about being bitten, for example. The effector subsystem would have no way to distinguish detection of being bitten from reflection of being bitten and so on the basis of that information alone could not act. Thus, useful messages between subsystems in C3 must include some kind of prescriptive information relevant to the subsystem in addition to the information about what is going on in some other subsystem. Now since the prescriptive information must be relevant to a particular subsystem, the message is only useful in that subsystem and so cannot usefully be posted in many subsystems.

Contrast subsystem interaction in C3 to what occurs in C4 when it posts the message of a second-order rule. In the example of the biting dog, C4's visual detector posts a message with a representation containing the information "Dog biting C4", like C3. In C4 this information activates a second-order rule which represents the entire representation of the dog biting C4. Thus, the second-order rule contains the information "C4 is being informed that C4's visual detector is detecting a dog biting C4". In more colloquial terms, the information is "C4 is seeing a dog biting C4". This information includes the context of the information about the biting dog and so is useful by itself. Thus, the information can be posted in every subsystem in the message part of a single second-order rule. In order for information to be useful, it must occur in some context. Second-order rules include that context with the information, whereas rules in C3 do not. Thus, the message part of second-order rules can be usefully posted in many subsystems, whereas messages containing representations in C3 cannot.

The advantage of being able to inform various subsystems simultaneously about what is occurring in a particular subsystem, by having a single second-order rule post its message in many subsystems at the same time, is that it allows C4 to disseminate information very quickly. The result is that certain information comes to dominate much

of the cognitive activities of the system. If a second-order rule contains a representation of a representation R , which is in turn a representation of x , then it contains information about x , meaning that it must include the schema by which the system encodes x . (Recall that in more developed systems, every subsystem uses the same encoding for the same information (4.1.3).) When the second-order rule posts its message in many subsystems, the schema corresponding to x will be passed into messages in each of these subsystems. Thus, within two steps of message posting, the schema will proliferate considerably, and information about x will become active throughout the system. As information about x is processed in various subsystems, other second-order rules can pass this information between the subsystems, resulting in further proliferation of the schema corresponding to x , so that information about x comes to dominate much of the cognitive activities of the entire system, including the second-order rules. C4 can use this effect to its advantage to learn and to act more quickly.

Having certain information about x come to dominate the cognitive activities of the system results in many messages with the same schema corresponding to x . This increases the likelihood that the system will determine a useful action concerning x in a fixed amount of time because there is a greater chance that a very strong rule relevant to the situation will be activated. In other words, strong rules containing the schema corresponding to x are rules that have led to rewarded actions concerning x in the past. The more rules containing the schema corresponding to x that are active, the more likely that a strong one will have its condition part matched and so lead to useful action. The more rules containing the schema corresponding to x that are active, the more possibilities for action concerning x the system will process and so the chances of rewarded actions are greater. This applies equally to cases in which C4 is determining an action to take and situations in which it is testing its rules with its predictor rules to learn better actions. Like C4, C3 can pass messages between its subsystems in an attempt to discover useful actions, but because of its second-order rules, C4 can do so much faster; i.e., C4 is likely to determine or learn useful actions much more quickly than C3.

5.1.3 Two Routes for Determining Action

The presence of second-order rules has an interesting influence on C4's cognitive structure. It effectively creates two routes to determining action operating in parallel with each other. C4 contains all of the cognitive architecture that C3 does, including rules that post messages between subsystems that are not second-order rules. We saw above that the message parts of these rules include prescriptive information along with the information being passed between the subsystems. It can happen in C4 that a message containing a representation R of x, in some subsystem S, activates a second-order rule and a first-order rule passing the information about x to another subsystem at the same time. Now the first-order rule is a part of some sequence leading to an action, so that sequence will be activated. While this is occurring, the second-order rule causes all of the subsystems to process information about x in order to determine some action. Thus, two parallel routes to determine an action are active in C4.

The first-order rule is simply a part of a sequence leading to an action, whereas the second-order rule initiates massively parallel activity in the system sending messages between its subsystems while each is processing information about x to determine an action. Because the sequence initiated by the first-order rule is likely to involve many fewer rules it will likely determine an action more quickly than the process initiated by the second-order rule. However, the two processes are not in total isolation from each other. The subsystem containing the sequence activated by the first-order rule, also has rules activated by the second-order rule active within it. These rules will compete with each other to post their messages on the message board of that subsystem. It is possible for the processes initiated by the second-order rule to block the action determined by the sequence initiated by the first-order rule by successfully posting a message in direct competition with a rule in that sequence. In this case, the processes initiated by the second-order rule will ultimately determine what action is taken by the system, despite the fact that the sequence initiated by the first-order rule is faster in determining an action.

The advantage of having the two routes of determining action is that C4 always has a useful action available to it, the useful actions it can determine as a C3 system. However, in addition to the C3 activities that occur in C4, it also has its second-order

rules, so if the activation of second-order rules happens to activate some rule stronger than those available to C4 qua C3, it can take some more useful action. If not, it acts as if it were a C3. Thus, C4 can do no worse than C3 in any given situation, but it will often do much better.

5.1.4 C4-Consciousness

We now have a sufficiently developed notion of a second-order rule to notice that active second-order rules provide C4 with many of the characteristics of consciousness we considered in the introduction. Firstly, a second-order rule contains content distinctive to human self-awareness. The messages contain information of the sort "I³⁴ am seeing a dog", and "I am thinking about an apple". A representation of the sort available to C3 provides a system with something like simple awareness, whereas a second-order rule provides something like simple awareness of a representation, or awareness of being aware. "[A] mental state is conscious if it accompanied by a suitable higher-order thought" (Rosenthal 1986, p.336). This is something like what Brook refers to as empirical self-awareness (Brook 1994, p.64); that is to say, the system has something like awareness of an entire representation and not just the contents of a representation. Also, if the content of a representation that a second-order rule represents happens to be the system itself, then C4 will have something like awareness of itself, which is Brook's notion of apperceptive self-awareness (ibid), the most distinctively human kind of awareness possible. So C4 has messages with content characteristic of both simple awareness and self-awareness, including apperceptive self-awareness, the latter in virtue of its second-order rules.

Secondly, we have seen how an active second-order rule results in some schema dominating much of the cognitive resources of the system in the sense that it is contained in many of the posted messages. What this means is that many of C4's "thoughts" contain information about the same thing. Of course, more than one second-order rule can be active at any one time so different schemata can dominate the system's resources at the

³⁴ A justification for why we can use "I" instead of C4 as the subject will be given in the discussion of the unity of C4-consciousness (5.2.2).

same time. However, since there are only a limited number of messages that can be posted on any one message board, the number of dominating schemata is restricted. Also, second-order rules are in competition with each other in every subsystem, so it is most likely that only the strongest ones will successfully post their messages. This increases the effect that only a few schemata will come to be contained in many messages at the same time. Thus, the system processes large amounts of information about a small number of things in each moment because of the actions of its second-order rules. In other words, only a small number of distinct things will usually occupy the "thoughts" of the system at any one time, which is another characteristic feature of consciousness.

A third feature distinctive of consciousness shared by second-order rules is that it is relatively slow in determining an action. As we saw in section 5.1.3, C4 can have two routes for determining an action active at the same time. One route is determined exclusively by first-order rules, whereas the other involves second-order rules. The two routes are very much like instinctive reactions and actions determined by conscious deliberation. Determining an action from first-order rules after the fashion of C3 is something like simple awareness, as we have seen above. The system takes an action because of something it is representing. The system is only "aware" of the object and not of itself responding to the object. On the other hand, action determined by second-order rules has the characteristics of more considered conscious action. The representations involved, being contained in second-order rules, include representations of the system as well as the object to which it is responding, much like conscious deliberation. This route of determining action is massively parallel, involving much of the system's cognitive activities. The price that the system must pay for involving significant amounts of its resources to determine an action and thereby having to process many more rules than the route involving only first-order rules is that like conscious deliberation, it is relatively slow.

Since having active second-order rules in a system gives it many features distinctive of consciousness, we make the following definition. A system with active representa-

tions and active second-order rules is C4-conscious.⁵⁵ Notice that we include representations in the definition of C4-consciousness to include the kind of "simple awareness" available to C3 among the C4-conscious phenomena. However, C3 is precluded from being C4 conscious by the conjunction, which makes active second-order rules a necessary condition of C4-consciousness. Of course, having an active second-order rule entails having an active representation to activate it so the definition is somewhat redundant, but the redundancy serves to highlight the inclusion of "simple awareness" in C4-conscious phenomena.

5.2 What is C4-Consciousness Like?

A somewhat counterintuitive consequence of C4-consciousness is that second-order rules depend on first-order representations for their activation, so C3-type actions are more basic in an important way. The evolution metaphor relating C3 and C4 is quite appropriate in this context. Until the system develops useful actions based on its first-order rules it cannot have strong second-order rules, since they will be unable to lead to rewarded actions. Thus, more "considered" actions depend on more reactive kinds of actions and they cannot even be initiated until the more basic actions have begun. First-order rules activate second-order rules, so C3-type actions must be in the process of being determined before second-order rules can become involved in determining action; the lower-order rules *cause* the higher-order rules to occur. "[W]e can explain a mental state's being conscious by hypothesizing that the mental state itself causes that higher-order thought to occur" (Rosenthal 1986, p.336). Despite being counterintuitive, this is exactly what Rosenthal suggests the relation between second-order rules and other rules should be. "It is not that *first* one goes into a higher-order state of self-observation, creating a higher-order thought, so that one can then report [act on] the lower-order thought by expressing the higher-order thought. It is rather that the second-order state (the better-informed state) comes to be *created* by the very process of framing the report [action]" (Dennett 1991, p.315).

⁵⁵ This term will allow us to avoid an inordinate number of scare quotes by having to refer to C4 as "conscious".

The dependence of second-order rules on first-order representations for their activation highlights the decentralized control of C4's actions. Which rules determine action depends on which ones can usurp which,⁵⁶ with control being totally distributed throughout the system; representations are representations only for those rules that they activate. C4's structure is certainly non-homuncular. The difficulty is that perhaps the most important single characteristic of consciousness is that it is unified. Can C4-consciousness, with its decentralized control and dependence of second-order rules on first-order representations for their activation, possess this feature of being unified? We turn now to this central issue, for regardless of the other characteristics of consciousness that C4-consciousness might share, this feature is essential.

5.2.1 *The Subject of C4-Consciousness*

... all the predicates of an internal sense refer to the *ego*, as a subject, and I cannot conceive myself as the predicate of any other subject. Hence completeness in the reference of the given concepts as predicates to a subject -- not merely an idea, but an object -- that is, the *absolute subject* itself, seems to be given in experience. But this expectation is disappointed. For the ego is not a concept, but the only indication of the object of the internal sense, so far as we cognize it by no further predicate. Consequently, it cannot be itself a predicate of any other thing; but just as little can it be a determinate concept of an absolute subject, but is, as in all other cases, only the reference of the internal phenomena to their unknown subject" (Kant 1783, pp.75-6)

Whatever is to serve as the subject of consciousness in any model cannot also be an object, since a subject qua object reintroduces the homunculus problem. The subject of consciousness is by definition "that which is conscious". Further inquiry into how it is conscious is to misunderstand the notion of a subject. However, a subject that is also an object invites just such inquiry. "How is it that this object is conscious?" The only way for such an inquiry to end and avoid an infinite regress is to have a subject that is not also an object.

Representations in C4 are representations to other rules in the system, but these rules are not C4-conscious. C4-consciousness *arises* when representations are represented, but what they are represented to is not what is C4-conscious. The subject of C4's

⁵⁶ The allusion to Dennett's Pandemonium model here is not accidental (Dennett 1991, p.237).

representations and hence of C4-consciousness is a represented subject and nothing more; it has no objective component. A second-order rule represents the subject of C4-consciousness -- the "to whom" the representation being represented is a representation -- with the schema corresponding to the entire system simply because the actual rules to which the representation is a representation could be scattered throughout the system, and so cannot be referred to specifically, as we saw in section 5.1.1. But the schema corresponding to the entire system is not *the thing* which is C4-conscious either. C4-consciousness has no subject that is also an object, so in objective terms C4-consciousness is subjectless. Instead, C4 has a represented subject, and in *every* second-order rule the schema used to represent the subject is the schema corresponding to the entire system. Thus, it is the entire system C4 that is the subject of C4-consciousness, but only in virtue of being represented subjectively, i.e., non-objectively.

5.2.2 The Unity of Consciousness at a Time

Most of our considerations of C4-consciousness thus far have concerned a single second-order rule. However, the definition of C4-consciousness allows for many second-order rules and representations to contribute to it at any one time; all that is required is that they are active. Thus, C4-consciousness is constituted of the content of every representation and second-order rule active at any one time superimposed on each other. This is a rather chaotic picture at first glance, particularly if C4-consciousness is to be unified in some way. Brook takes the position that unity requires mental representations to be "tied together" in some way (Brook 1994). However, Seager responds to Brook with the claim that mere simultaneity is sufficient for unity. "Of course, certain contents will achieve sufficient "cerebral celebrity" to make it past the (fuzzy) line demarcating the conscious from the non-conscious as a group. They will all be conscious together, so to speak, but they won't be necessarily unified in any other way" (Seager unpublished). So if Seager is right, C4-consciousness has all of the unity it needs. Unfortunately, simultaneity is not sufficient for unity.

If X has one thought and Y has some other thought at the same time, it is not necessary that these thoughts be bound together in any way, particularly if X and Y are

not related to each other in any obvious way. In fact, in such a case we would want to say that the thoughts are unrelated. But it may well be that X and Y each became conscious of their respective thoughts at the same time. Both thoughts passed the "fuzzy line" between the conscious and non-conscious at the same moment. So in Seager's account, these thoughts should be as unified in conscious as any thoughts could be, which is rather strange for two thoughts that are unrelated. Notice that what these thoughts did not do is pass this fuzzy line as a group; but whatever defines a group is a uniting feature that "ties them all together". Of course, this is not really a fair criticism of Seager because the thoughts did not cross the same fuzzy line. So what if they had? What if X and Y were the same person? How would the picture be any different? What if the thoughts originated in different subsystems of the same person's brain; how could they be any more unified than if they originated in different people's brains?

Clearly there is something important to distinguish between one person having two thoughts and two people each having one thought at a given moment. In the first case the thoughts will be unified in a way that they cannot be in the second case. Now appealing to a fuzzy line "demarcating the conscious from the non-conscious" as opposed to two lines is not adequate to distinguish between the cases, because it already presupposes the answer to the very question we are asking. A fuzzy line is associated with consciousness. Being able to distinguish one fuzzy line from two presupposes being able to quantify over consciousness, which requires individuating consciousness, which in turn entails some kind of unity of each individual consciousness. Thus, simultaneity can only function as a sufficient condition for unity of consciousness if there is some independent determination of that unity. Since simultaneity of mental contents alone cannot serve as a criterion for differentiating between two clearly different cases of unified consciousness, something else must be required, something "tying mental states together".

Returning to C4 we are left with the problem of "tying together" all of the contents in C4-consciousness if it is to have the kind of unity characteristic of human consciousness. Note that despite the fact that we have already established that there is a single subject of C4-consciousness, this is not enough to show that C4-consciousness is unified.

The content in C4 consciousness must be more than just content for the same subject if it is to be made conscious in a way resembling human consciousness of content. The content of our consciousness remains individuated but it stands in relation to all of the other content. We are able to compare things in our mind. C4 is able to do something similar, however, by combining representations in second-order rules that include relational information. In other words, distinct representations are represented in relation to each other in C4's second-order rules. Thus, the representations remain individuated, but are "tied together" in virtue of being represented in relation to each other. This unifies the objects in C4's consciousness into a sort of "global object" of which the system is C4-conscious.

Total unity of C4-consciousness requires more than just a single subject, even when the content is unified. The reason for this is that different subsystems are *active* in determining different contents in C4-consciousness. Even though the content may be unified content for the same subject, it has a myriad of sources. A system can have no sense of "I" unless the subject of consciousness is also represented as the active agent. Now recall that every second-order rule includes information indicating which C4 subsystem is the origin of certain content. Thus, the content of C4's second-order rules is of the form "C4's visual detector is representing a dog to C4" for example. In particular, C4 is identified in each case, since the subsystems are referenced as parts of C4. So C4 is both the active agent, qua object, and the subject of C4-consciousness, qua represented subject. That is to say, the content of C4-consciousness is that any action taken, including internal ones like thoughts, is done by the subject of C4-consciousness; hence, the schema corresponding to the system, which is used in every case, takes the role of "I" in the system's C4-consciousness; "I" is C4's unity of consciousness at a time.

A final point concerning C4's unity of C4-consciousness concerns its first-order representations. First-order representations do not contain the same content as second-order rules. In fact, they have neither an active agent nor a subject included in their content, and so do not stand in any relation to the system's unity of C4-consciousness. But this is exactly what should be the case. Their content consists of things of which C4 has

something like simple awareness, which by definition is not in relation to its "I". However, whenever C4 reflects on the content of a first-order representation it does so in virtue of a second-order rule, thereby putting the content in relation to its unity of C4-consciousness. Thus, any content involved in more than just instinct-like actions is represented in relation to C4's "I", as is the case with human consciousness.

5.2.3 Memory and Unity of Consciousness Across Time

As we have seen above, the only memory a classifier system has is in its existing rules. This need not be the case in C4, however. It is possible that it could "store" a representation as it occurs within the system by making it part of a message that also included some kind of chronological information such as a date. In other words, the message would contain the information that a certain representation occurred at that time.⁵⁷ Like all other rules, classifiers with message parts containing representations that previously occurred in the system could be activated by any message matching their condition part. Since representations occurring together in the system will often do so under similar conditions, it will be the case that similar conditions will activate rules containing them, so that memories would tend to be recalled in clusters and could often activate each other. Memories could take an active role in determining what actions C4 takes because the messages posted containing representations that had previously occurred in the system could activate other rules and so become part of a sequence leading to action.

Stored representations could be either first or second-order representations. Activating a rule containing a previously occurring first-order representation would make the system C4-conscious of the content of the representation. Activating a second-order representation that previously occurred, on the other hand, would make the system C4-conscious of an entire representation, including its reference and function. Now, as with any other second-order rule, the representation being recalled would represent C4 as the active agent and the subject of the representation. But since these agree with the active

⁵⁷ Exactly how the system would experience time as a result of dating its messages is an interesting and important question but it is well beyond the scope of this essay.

agent and subject in every second-order rule active when the memory is activated, the system will be C4-conscious of its "I" as being subject of its past C4-consciousness. In other words, the system represents the "I" in the past in the same way it represents the "I" in the present, so it identifies them in C4-consciousness. This is C4's unity of C4-consciousness over time.

5.3 A Walk in the Park with C4

I sometimes like to end the day the way I start it, by taking my dogs for a walk. Let's suppose we are back in the dog park, only this time C4 has come along for the walk. What will it be like for C4?

5.3.1 Abstraction: A Final Look

In the last chapter (4.1.4), we stipulated that C3 would have categories corresponding to the schemata that were the content of representations. In the context of C3, this is a somewhat arbitrary stipulation, since many schemata could become efficacious that are not the sense of a representation. However, in C4 there is an important difference between schemata that are the content of representations and other schemata active in the system. The difference is that the information these schemata encode corresponds to the objects of C4-consciousness. Recall that a system is C4-conscious when it has active representations and second-order rules (5.1.4); they exclusively determine its content. Thus, the only things of which a system can be C4-conscious are what its representations are about, i.e., the sense of its representations.⁵⁸ It follows that since categories are categories of *things*, the only categories the system has are those determined by the schemata that are the sense of some representations. Now since the system has C4-consciousness of these categories, the categories are something like abstract concepts for C4. In other words, the only "awareness" C4 has is of the contents of its representations, so these contents determine the "concepts" by which it can categorize information.

⁵⁸ Note: that while second-order rules do contain information about the function and reference of the representations they represent, these are simply the system and its detectors, whose schemata can also occur in the sense of a representation, so the content of C4's representations is equivalent to the things of which it is C4-conscious.

So the complete description of abstraction in C4 is as follows. C4 extracts information from a message posted in one of its subsystems by passing some schema contained in the message into other messages. In one of the messages into which the schema is passed, the source of that schema -- the subsystem in which it originated -- is indicated by a tag, making the message a representation of the information encoded by the schema. At this point the system has "simple awareness" of whatever corresponds to the schema; the information is in the system's C4-consciousness. The information can then be passed into a second-order representation so that the system has a representation of representing the information, which is something like self-awareness, a representation of the system in relation to the information. Once the schema becomes part of a representation and so enters C4-consciousness, the information encoded by it is categorized according to all of the subschemata it contains, including itself. Information of which the system is C4-conscious belongs to each of the categories defined by subschemata contained in the schema encoding it, since it is represented as having the defining characteristics of that category. Thus, a particular instance of some thing is categorized by being represented in the same way as the other members of the category.

So what happens when C4 comes walking in the dog park? Well, C4 has seen dogs before so its rules have co-adapted in such a way that the schema posted in all of its detectors when it encounters a dog can be extracted and passed through messages to become the content of a representation. When seven or eight dogs begin running and barking in a pack, C4 forms representations of each of them, and each of these representations includes the same schema, which is always posted by its detectors when it encounters dogs. Now since each of the dogs is represented, C4 is C4-conscious of each of the dogs. Furthermore, it is C4-conscious of them *as* dogs because they are all represented as such in virtue of containing the schema corresponding to "dog" in their individual representations. Also, the representations of each dog activate second-order rules, so that the system becomes C4-conscious of representing the dogs. Second-order rules representing representations of the dogs in its visual, auditory, olfactory, and internal subsystems make the system C4-conscious of "seeing", "hearing", "smelling".

and "thinking" about the dogs. Also, because of the unity of C4-consciousness, the system represents itself, C4, as doing so in each case. That is, the system represents itself as the subject of C4-consciousness and the active agent. C4 has successfully abstracted the category "dog".

5.3.2 Making Predictions

C4 can be C4-conscious of the predictions its predictor rules make because the message parts of its predictor rules contain representations. This follows simply because a predictor rule tags a detector as the source of certain information, thereby indicating a particular instance of the information and so making reference to it. Clearly the message in the predictor rule contains information, and the sense and reference of the message are sense and reference for all of the rules they activate. Hence the message part of a predictor rule is a representation. C4 makes predictions about its environment when large collections of predictor rules make individual predictions at the same time (4.2). If the conditions facing the system are novel, then a novel collection of predictor rules will be activated and so the system will predict a novel occurrence in the environment. Now since predictor rules contain representations in their messages, those representations can be represented by second-order rules; and because of the unity of objects in C4-consciousness, a large number of co-active second-order rules representing predictions will result in the system being C4-conscious of making a prediction about its environment. It is important to notice that if the system does not possess strong predictor rules it cannot make good predictions, and developing strong predictor rules requires numerous encounters with the environment. Baby C4's, like baby humans, are notoriously bad at making predictions.

Our C4 is an adult system, and so has developed good predictor rules. When we walk towards the wooded area, C4 visually detects squirrels, for which it has an abstract concept, which activates a number of predictor rules. In particular, the conditions activating the predictor rules also contain active information about dogs. When all of the conditions of "dog looking at squirrel" are satisfied, a large number of predictor rules become active and they make predictions that certain messages will be posted by particu-

lar detectors. Second-order rules representing these predictions make the system C4-conscious of predicting an environmental event. In this case, the predictions taken together as unified in C4-consciousness contain representations of my dogs running after the squirrels, which they do, but not catching them, which they do not. That is to say, C4 successfully predicts my dogs' actions.

5.3.3 Heading for Home

While the dogs are running after the squirrels a branch falls from a tree towards C4. C4's visual detectors post messages containing information about the falling branch. Some of this information is extracted and passed through to other messages, where information tagging its source is included so that C4 has a representation of the falling branch. The representation activates a second-order rule, and is simultaneously passed through to a message in an effector subsystem that also includes information prescribing the action "move". Before the massively parallel activities initiated by the second-order rule determine an action, the sequence of first-order rules is successful in posting all of its messages because none of the rules activated by the second-order rule is strong enough to outbid the first-order rules; so C4 moves out of the way of the branch. It is interesting that C4's second-order rules could not determine an action fast enough to allow it to move before being hit.

When we leave the woods C4 sees a dog by itself. C4 also has a representation of cold through its tactile detectors. These conditions together activate rules containing representations of cold and sick dogs, which in turn activate representations of the nearby animal hospital. These conditions activate C4's effectors in such a way that it acts to catch the dog. After several unsuccessful attempts, unrewarded actions, rules in C4 successfully post messages by out-bidding the "dog catching" rules, thereby changing the actions of the system, so it ceases its efforts to catch the dog and resumes its walk. As it walks, however, many of its messages contain content about the loose dog. Some of these representations are represented by second-order rules so that soon thoughts of loose dogs and actions to catch them dominate most of the system's C4-consciousness. The system activates its predictor rules in lieu of actual environmental conditions, testing various

actions in its "mind". Eventually a representation of dog cookies is activated by the numerous instances of the schema corresponding to dog active in the system. When the action of offering a dog cookie is tested, the predictor rules as represented by second-order rules make the system C4-conscious of predicting that the dog would come, which in these conditions is rewarded action for that system. Thus, C4 develops the plan to offer dog cookies to stray dogs, and hence to carry some on walks.

Finally as we turn for home, C4 detects the dog that was running loose now on a leash walking beside a person. This new information outbids competing rules about dogs but does not activate further strong rules with messages containing information about dogs. Messages containing information about things other than dogs are then successfully posted and C4's "thoughts" turn to other things. It has been a rather *extra-ordinary* day after all.

5.4 Conclusion

Clearly, I have not produced a mechanistic model of consciousness. Many extremely important aspects of consciousness have not even been considered. For example, I have said nothing about emotions, the important relation between consciousness and language, our conscious sense of time, qualia, or volition. Nonetheless, I have been able to develop a model that does display some important features of consciousness. Its rules contain a special kind of content much like simple awareness and self-awareness, it is relatively slow in determining actions, it is dominated by only a few categories at any one time, and its information is unified. Mechanistic analogues of abstraction, representation, single-instance learning, and prediction are developed. Furthermore, by adopting a bottom up approach, I have developed this model without making any appeal to an undischarged homunculus -- or indeed a homunculus of any kind. The subsystems that are specialized for certain tasks have complete control over those tasks, with no central organizer between subsystems or within them. Everything the system does, it does mechanistically. The strength of a model like this is that it does enough of what we do to hold some promise of scaling up without inadvertently introducing unreduced cognitive mysteries such as homunculi and control centres. Where it falls short is in providing

mechanisms within the subsystems, but those are the objects of study in neuroscience and other disciplines. It is entirely plausible that progress can be made in better understanding how processes within the subsystems work. My model requires looking "downward" to the mechanisms rather than "upward" to the central controller in order to understand consciousness, and I think that by itself makes it a step in the right direction.

BIBLIOGRAPHY

- Brook, J.A. 1994. *Kant and the Mind*. New York: Cambridge University Press.
- Churchland, P. 1988. *Matter and Consciousness*. Cambridge: Bradford/MIT Press.
- Chomsky, N. 1959. "A Review of B.F. Skinner's *Verbal Behaviour*" in *The Structure of Language Readings in the Philosophy of Language*, edited by Fodor, J. and Katz, J. 1964. Englewood Cliffs, NJ: Prentice-Hall Inc..
- Davis, L. 1991. *Handbook of Genetic Algorithms*. New York: Van Nostrand Reinhold.
- Dahlbom, B. 1993. *Dennett and His Critics*. Cambridge: Blackwell.
- Dennett, D. 1978. *Brainstorms: Philosophical Essays on Mind and Psychology*. Cambridge: Bradford/MIT Press.
- Dennett, D. 1987. *The Intentional Stance*. Cambridge: Bradford/MIT Press.
- Dennett, D. 1991. *Consciousness Explained*. Boston: Little Brown and Company.
- Dretske, F. 1988. *Explaining Behaviour: Reasons in a World of Causes*. Cambridge: Bradford/MIT Press.
- Fodor, J. 1981. *Representations: Philosophical Essays on the Foundations of Cognitive Science*. Cambridge: Bradford/MIT Press.
- Fodor, J. 1990. *A Theory of Content and Other Essays*. Cambridge: Bradford/MIT Press.
- Fodor, J. and Lepore, E. 1993. "Is Intentional Ascription Intrinsically Normative?" in Dahlbom 1993.
- Goldberg, D. 1989. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading MA: Addison-Wesley Publishing Company, Inc..
- Holland, J. 1975. *Adaptation in Natural and Artificial Systems*. Ann Arbor: The University of Michigan Press.
- Holland, J. 1992. "Genetic Algorithms" in *Scientific American*, July 1992, pp.66-72.
- Holland, J., Holyoak, K., Nisbett, R., and Thagzrd, P. 1987. *Induction: Processes of Inference, Learning, and Discovery*. Cambridge, MA: The MIT Press.
- Horgan, J. 1994. "Can Science Explain Consciousness?" in *Scientific American*, July 1994, pp.88-94.
- Kant, I. 1783. *Prolegomena to Any Future Metaphysics*. Carus, P. (trans.), Ellington, J.W. (revised) 1977. Indianapolis: Hackett.
- Penrose, R. 1989. *The Emperor's New Mind: Concerning Computers, Minds, and the Laws of Physics*. London: Vintage/Oxford University Press.
- Rosenthal, D. 1986. "Two Concepts of Consciousness" in *Philosophical Studies*, vol. 49, pp.329-359.
- Seager, W. (unpublished, 1994). "Materialism, Apperception, and the Unity of Consciousness".

- Sober, E. 1985. "Putting the Function Back into Functionalism" in *Mind and Cognition: A Reader*. Lycan, W. (edited) 1990. Cambridge: Basil Blackwell.
- Stillings, A., Feinstein, M., Garfield, J., Rissland, E., Rosenbaum, D., Weisler, S., Baker-Ward, L. 1987. *Cognitive Science: An Introduction*. Cambridge: Bradford/MIT Press.

END

3 | 0 - 0 5 - 9 5

FIN