1

MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS
STANDARD REFERENCE MATERIAL 1010a
(ANSI and ISO TEST CHART No. 2)

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

Vostra libreria - Votre bibliothèque

Ваша библиотека - Notre bibliothèque

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

COMPUTATIONAL EVOLUTIONARY EPISTEMOLOGY

by

DWIGHT L. DEUGO, B.C.S., M.C.S.

A thesis submitted to
the Faculty of Graduate Studies and Research
in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

Ottawa-Carleton Institute for Computer Science
School of Computer Science
Carleton University
Ottawa, Ontario
June 3, 1993

© copyright
1993, Dwight L. Deugo

National Library
of Canada

Bibliothèque nationale
du Canada

Acquisitions and
Bibliographic Services Branch

Direction des acquisitions et
des services bibliographiques

395 Wellington Street
Ottawa, Ontario
K1A 0N4

395, 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 accordé une licence irrévocable et non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de sa thèse de quelque manière et sous quelque forme que ce soit pour mettre des exemplaires de cette thèse à la disposition des personnes intéressées.

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 propriété du droit d'auteur qui protège sa thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

ISBN 0-315-89762-7

Canada

Name Dwight Deugo

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.

Computer science

SUBJECT TERM

0984

U·M·I

SUBJECT CODE

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
General 0679
Ancient 0289
Linguistics 0290
Modern 0291
Literature
General 0401
Classical 0294
Comparative 0295
Medieval 0297
Modern 0298
African 0316
American 0591
Asian 0305
Canadian (English) 0352
Canadian (French) 0355
English 0593
Germanic 0311
Latin American 0312
Middle Eastern 0315
Romance 0313
Slavic and East European 0314

PHILOSOPHY, RELIGION AND THEOLOGY

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

SOCIAL SCIENCES

American Studies 0323
Anthropology
Archaeology 0324
Cultural 0326
Physical 0327
Business Administration
General 0310
Accounting 0272
Banking 0770
Management 0454
Marketing 0338
Canadian Studies 0385
Economics
General 0501
Agricultural 0503
Commerce Business 0505
Finance 0508
History 0509
Labor 0510
Theory 0511
Folklore 0358
Geography 0366
Gerontology 0351
History
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
General 0615
International Law and Relations 0616
Public Administration 0617
Recreation 0814
Social Work 0452
Sociology
General 0626
Criminology and Penology 0627
Demography 0938
Ethnic and Racial Studies 0631
Individual and Family Studies 0628
Industrial and Labor Relations 0629
Public and Social Welfare 0630
Social Structure and Development 0700
Theory and Methods 0344
Transportation 0709
Urban and Regional Planning 0999
Women's Studies 0453

THE SCIENCES AND ENGINEERING

BIOLOGICAL SCIENCES

Agriculture
General 0473
Agronomy 0285
Animal Culture and Nutrition 0475
Animal Pathology 0476
Food Science and Technology 0359
Forestry and Wildlife 0478
Plant Culture 0479
Plant Pathology 0480
Plant Physiology 0817
Range Management 0777
Wood Technology 0746
Biology
General 0306
Anatomy 0287
Biostatistics 0308
Botany 0309
Cell 0379
Ecology 0329
Entomology 0353
Genetics 0369
Limnology 0793
Microbiology 0410
Molecular 0307
Neuroscience 0317
Oceanography 0416
Physiology 0433
Radiation 0821
Veterinary Science 0778
Zoology 0472
Biophysics
General 0786
Medical 0760

EARTH SCIENCES

Biogeochemistry 0425
Geochemistry 0996

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

HEALTH AND ENVIRONMENTAL SCIENCES

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

Speech Pathology 0460
Toxicology 0383
Home Economics 0386

PHYSICAL SCIENCES

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

Applied Sciences

Applied Mechanics 0346
Computer Science 0984

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

COMPUTATIONAL EVOLUTIONARY EPISTEMOLOGY

submitted by

DWIGHT L. DEUGO, B.C.S., M.C.S.

in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

Director, School of Computer Science

Thesis Supervisor

External Examiner

Carleton University

November 30, 1993

Abstract

In this thesis we develop Computational Evolutionary Epistemology (CompEE), a biologically motivated approach to cognitive modeling and machine learning that uses abstractions of known biological mechanisms and simplified Darwinian processes, for the computational study of the evolution of individual learning and knowledge on a computer.

In CompEE learning, in the sense of evolutionary changes to an individual's internal organization that improve its performance, occurs at multiple levels, each of which uses a different structure to encode its symbolic knowledge. Evolution is used as a universal process for knowledge gain at each level, manipulating hierarchies to evolve complex structures including knowledge. At CompEE's highest level a new model of reasoning, called Evolution-Based Reasoning, produces reliable knowledge by applying various 'genetic operators' to the sorts of symbolic structures that are used in traditional Artificial Intelligence systems. At its lowest level, CompEE begins to tackle the 'preprogramming' problem of Machine Learning - the hand-coding of knowledge and control parameters.

CompEE is concerned with answering questions like the following: is the very process of thinking an evolutionary one, i.e. is the perceived sequential train of thoughts the outcome of Darwinian selection, where pieces of thoughts are put together and compared with memories as to their appropriateness in the current situation, and the best 'fitting' one selected as the next part of the 'train of thought'? *Does the brain operate as an Evolutionary Machine?*

Some of the ideas underlying CompEE were stimulated by research done in Evolutionary Epistemology. Many of Evolutionary Epistemology's current arguments are philosophical in nature. This is a suitable format for a philosophical audience. However, for an audience of people who want to make use of the ideas, those in Artificial Intelligence, Machine Learning and Cognitive Science, a computational approach demonstrating the viability of such ideas is required. This thesis attempts to do just that: a computational approach to Evolutionary Epistemology.

Acknowledgements

When I started my Ph.D. I knew that I would have to rely on several people to help me complete it. However, what I didn't realize then was just how much time and effort I would require from them.

At the head of this list is my advisor and friend Professor Franz Oppacher. Franz's repeated ideas, discussions, suggestions, instructions, and insights provided me with the focus, determination, and knowledge to continue. Both on a personal and professional level, I have accumulated a debt which I am sure I will never be able to repay.

Next, is my wife Sue, without whose patience and understanding none of this would have been possible.

Since it is important to talk with other people about your ideas, not just to inform others about them, but to get them straight in your own mind, I thank Una-May O'Reilly for listening to me no matter how vague or premature my ideas seemed.

Finally I want to thank my kids Ryan, Samantha and Jake for letting me work even when I should have been playing with them.

I also gratefully acknowledge the financial support from Bell-Northern Research, the National Science and Engineering Research Council, Carleton University's Faculty of Graduate Studies and the Ontario Government.

Table of Contents

Abstract	iii
Acknowledgements	iv
List of Tables	ix
List of Illustrations	xi
List of Appendices	xiii
1 Introduction	1
1.1 Current Approaches to Intelligence	2
1.2 CompEE's Approach to Intelligence.....	9
1.3 EE Predecessors	13
1.4 Organization	15
1.5 Chapter Summary.....	17
2 The Origin and Evolution of Evolutionary Thought	18
2.1 Evolution.....	21
2.2 Evolutionary Epistemology Programs.....	25
2.3 Objections to Evolutionary Epistemology.....	33

2.4	Conclusion.....	37
2.5	Chapter Summary.....	38
3	Computational Evolutionary Epistemology	40
3.1	What is Learning?.....	42
3.2	What is ComPEE?	43
3.3	ComPEE's SimOrg	49
3.4	Chapter Summary.....	53
4	The Genotypic Level	54
4.1	The Evolution of Hierarchies.....	55
4.1.1	Hierarchical Chromosomes.....	57
4.1.1.1	Sequential Genetic Algorithm	62
4.1.1.2	Random Sequential Genetic Algorithm	64
4.1.1.3	Tree Genetic Algorithm	66
4.1.1.4	Subtree Genetic Algorithm	72
4.1.2	Review of GA Organizational Strategies.....	77
4.2	Genome	79
4.2.1	Chromosome Hierarchy.....	81
4.2.2	Motivational Drive Hierarchy.....	86
4.3	Evolution and Fitness of the Genome	88
4.4	Chapter Summary.....	90
5	The Phenotypic Level	93
5.1	Motivational System.....	95

5.2	Stimulus System	97
5.3	Memory System.....	98
5.3.1	Schema Types.....	102
5.4	Problem-Solving System	106
5.4.1	The Different Views of a Schema and its Response	109
5.5	Evolution and Fitness of a SimOrg	112
5.6	The Control Sequence.....	113
5.7	Chapter Summary.....	118
6	The Cognitive Level	121
6.1	Evolution-Based Reasoning	123
6.2	Schema Fitness.....	130
6.3	Schema Operators.....	133
6.4	Chapter Summary.....	138
7	Experiments	139
7.1	The Environment.....	141
7.2	Plentiful Environment	145
7.2.1	Species Learning.....	147
7.2.2	Individual Learning.....	157
7.2.3	SimOrgs With Sensory Deficiencies.....	166
7.2.4	Alternate Environmental Objects	172
7.3	Spacious Environment	178
7.4	Hazardous Environment	183
7.5	Chapter Summary.....	187

8 Summary and Conclusion	189
8.1 CompEE's Achievements.....	190
8.2 Future Research Direction	196
Appendices	199
Bibliography	203

List of Tables

1.1	Summary of Different Approaches to Intelligence.....	8
1.2	Comparison of CompEE with other Approaches to Intelligence.....	13
4.1	Gene Fitness Contribution.....	60
7.1	Gene Ranges	140
7.2	Environmental Object Features.....	141
7.3	Movement Feedback	143
7.4	Drinking Feedback	143
7.5	Eating Feedback	144
7.6	Sniffing Feedback	144
7.7	Reproduction Feedback.....	145
7.8	Template and Condition Dropping Gene Trends.....	149
7.9	Long and Short-Term Memory Trends	151
7.10	Final Drive Levels	153
7.11	Population Age and Death Trends	156
7.12	Positive and Negative Schema Counters.....	158
7.13	Evolved verses Randomly Produced Schemata.....	160
7.14	Template and Condition Dropping Gene Trends.....	167
7.15	Long and Short-Term Memory Trends	168
7.16	Population Age and Death Trends	170

7.17	Positive and Negative Schema Counters.....	171
7.18	Evolved verses Randomly Produced Schemata.....	171
7.19	Template and Condition Dropping Gene Trends.....	173
7.20	Long and Short-Term Memory Trends	174
7.21	Population Age and Death Trends	175
7.22	Positive and Negative Schema Counters.....	176
7.23	Evolved verses Randomly Produced Schemata.....	177

List of Illustrations

1.1	Simplified Case-Based Reasoning Components.....	4
2.1	Evolutionary Epistemology Program Connections	32
3.1	SimOrg Hierarchy	51
4.1	Tree Chromosome.....	61
4.2	Duplicate Node Removal	69
4.3	Duplicate Root Node Removal.....	69
4.4	Missing Node Addition	70
4.5	Complete Chromosome Produced by the STGA	75
4.6	Complete Trees Represented by One Subtree	78
4.7	Gene Ordering Dependent GAs.....	79
5.1	A Drive.....	95
5.2	A Schema.....	99
5.3	Plan Schema.....	104
5.4	Complex Schema	105
5.5	Final Schema.....	105
5.6	External View of a Complex Plan Schema	109
5.7	Simple and Complex Analogy Schemata	111
6.1	Mutation Operator Applied to a Simple Schema	133
6.2	Mutation Operator Applied to a Complex Schema.....	134

6.3	Crossover Operator Applied to a Simple Schema.....	135
6.4	Crossover Operator Applied to a Complex Schema	135
6.5	Split Operator Applied to a Simple Schema	136
6.6	Split Operator Applied to a Complex Schema.....	137
6.7	Join Operator Applied to a Simple and a Complex Schema	137
7.1	Plentiful Environment.....	146
7.2	Template and Condition Dropping Genes That Are Set To True in Test Run #1	147
7.3	Sum of the Long and Short-Term Memory Sizes.....	151
7.4	Expected Motivational Drive Hierarchy	152
7.5	Average Motivational Drive Hierarchy.....	153
7.6	Age of Population and Age of SimOrg's Death.....	157
7.7	Positive and Negative Schemata Counters of Test Run #10.....	159
7.8	Evolved verses Randomly Produced Schemata in.....	162
7.9	Spacious Environment	179
7.10	Positive and Negative Schema Counters.....	181
7.11	Hazardous Environment.....	184
7.12	Evolved versus Random Actions in Hazardous Environment.....	186
7.13	Evolved versus Random Actions in Hazardous Environment with Infinite Life SimOrgs	186

List of Appendices

A	Glossary	199
B	Acronyms.....	201

Chapter 1

Introduction

Since knowledge is a product of our biology, and since our biology is a product of evolution, knowledge can be explained in terms of evolution. Our approach to this conception of Evolutionary Epistemology (EE) is called Computational Evolutionary Epistemology (CompEE).

CompEE is a biologically motivated approach to cognitive modeling and machine learning that uses abstractions of known biological mechanisms and simplified Darwinian processes, for the computational study of the evolution of individual learning and knowledge on a computer. It has five goals:

- 1) To show that learning, in the sense of evolutionary changes to an individual's internal organization that improve its performance, occurs at multiple levels, each of which uses a different structure to encode its symbolic knowledge.
- 2) To show that evolution is used as a universal process for knowledge gain.
- 3) To show how to use hierarchies for evolving complex structures, including knowledge.

- 4) To show the production of reliable, symbolic knowledge using evolution; thereby, forming a new model of reasoning called Evolution-Based Reasoning (EBR).
- 5) To begin to tackle the 'preprogramming' problem of Machine Learning (ML) - the hand-coding of knowledge and control parameters .

CompEE follows the holistic, synthetic approaches to intelligence of Animats and Artificial Life (AL). As in Genetic Algorithms (GAs) and Classifier Systems (CSs), CompEE uses the process of evolution for manipulating its structures; however, in contrast to these approaches, CompEE works with the symbolic structures of traditional artificial intelligence (AI) systems. CompEE is ultimately interested in evolution that takes place on a time scale of milliseconds in an organism's brain. CompEE is concerned with answering questions like the following: is the very process of thinking an evolutionary one, i.e. is the perceived sequential train of thoughts the outcome of Darwinian selection, where pieces of thoughts are put together and compared with memories as to their appropriateness in the current situation, and the best 'fitting' one selected as the next part of the 'train of thought'?

Does the brain operate as an Evolutionary Machine?

1.1 Current Approaches to Intelligence

Traditional AI has proposed many different models of specific human abilities, such as planning [Hammond, 1989], searching [Nilsson, 1980], inductive learning [Holland *et al.*, 1986], and natural language processing [Winograd, 1983] to name but a few. Although powerful and transparent within specific domains, 'symbolic' AI systems suffer from brittleness outside narrowly defined domains [Holland, 1986; Holland, *et al.*, 1986]. As Wilson points out [Wilson, 1991], the brittleness of many AI systems results from their addressing only 'isolated competences':

- AI systems often ignore the fact that they must work in sensory environments and satisfy multiple conflicting goals.
- AI systems are often more concerned with algorithmic processes like search and exact reasoning than with natural abilities like adaptation.

One notable symbolic AI system that attempts to be more flexible is a case-based reasoning (CBR) system. A CBR system is designed to learn from experience and to reuse what has been learned instead of constructing new solutions from scratch [Hammond, 1989; Kolodner *et al.*, 1987; Kolodner, 1988]. A simplified view of CBR, shown in figure 1.1. contains four components: retrieval, adaptation, evaluation, and storage. The retrieval component locates the most appropriate case in memory to adapt and solve a given problem. The adaptation component alters the differences between the best case and the current problem to return the best possible solution. The evaluation component looks at the adapted case's outcome and performance - was the case's solution good or bad - for the purpose of updating or creating a new case in case memory. The storage component adds the new case with the appropriate indices, made from possible features, so the case can be retrieved in subsequent sessions.

CBR systems compare favorably with other 'symbolic' approaches such as rule-based expert systems or planners that always work from first principles. But their potential as a general approach to machine learning is diminished by their reliance on unchanging adaptation rules which tend to produce repetitive solutions, and by their current inability to maintain a library of nonredundant cases. These shortcomings are exacerbated when CBR is not just used in situations, such as expert systems tasks, in which occasional learning is treated as a welcome side effect, but when learning itself becomes the primary goal.

Figure 1.1 Simplified Case-Based Reasoning Components

The main mode of reasoning in CBR, i.e., to solve problems by ‘tweaking’ the most similar instances of past solutions, tends to proliferate repetitive solutions, and to fill memory with clusters of cases that differ from one another in only minor details. Even strong, non syntactic adaptive methods will make similar adaptations of a case whenever they are invoked. These ad-hoc tweaking rules tend to further diminish the effectiveness of memory by increasing redundancy and by precluding ‘creative’ solutions.

CBR learning is not merely conservative in the sense of accumulating similar cases but also in the following sense: since it is entirely failure-driven, it lacks a facility to opportunistically look for better solutions, for novel arrangements of available information, in the absence of out and out failures. Several researchers [Schank 1986; Thagard, 1988]

have deplored this lack of 'creativity' in failure-driven CBR systems. Schank has proposed [Schank, 1986, 140ff.] various ways to enhance the 'creativity' of CBR systems, such as the intentional misapplication of 'explanation patterns' (which are themselves fixed and prespecified) [ibid, 109ff.] or the generation of apparently irrelevant indices for CBR search.

The use of creativity-enhancing methods proposed by Schank will occasionally result in useful novel categorizations and 'creative' alterations of cases. In the long run, however, these methods, since they are essentially random stabs at novelty, will lead to a deterioration of the overall quality of case memory: memory will quickly become cluttered with useless cases, or, when the randomly generated cases are allowed to displace existing ones, successful past solutions may get lost.

'Subsymbolic' systems such as Neural Networks (NN) [Hinton, 1990a, 1990b; Judd, 1990] and GAs [De Jong, 1990; Goldberg, 1989; Davis, 1987; Holland, 1975; Grefenstette, 1988; Wilson, 1989] show some characteristic advantages such as robustness, capability of handling contradictions, and fault tolerance, but, because of their lack of transparency and inability to handle chains of inferences, are not yet convincing in traditional AI applications like expert systems or planners.

'Subsymbolic' approaches other than GAs, e.g. connectionist systems, have so far proven weak in domains requiring high level knowledge processing. In cognitively complex domains, it is important to be able to use previously acquired domain knowledge. The use of such knowledge may not just improve performance but may actually make the difference between satisfactory and unacceptable results [Fogelmann-Soulie, *et al.*, 1990]. GAs naturally represent and profit from previously acquired knowledge ('building block hypothesis', [Goldberg 1989, 41ff.]), and the introduction of rules into a CS by a knowledge engineer is straightforward. This contrasts sharply with connectionist systems,

in which the introduction of domain knowledge requires ad-hoc 'connections engineering', i.e. the careful wiring of the network architecture.

In contrast to the top down approaches to AI, the AL [Langton, *et al.*, 1992] and Animat [Wilson, 1991] approaches to intelligence start from below. 'Artificial Life involves the realization of lifelike behavior on the part of man-made systems consisting of populations of semi-autonomous entities whose local interactions with one another are governed by a set of simple rules' [Langton, 1989, xxiiff.]. 'The Animat approach advocates maintaining the holism of the situation of real animals in real environments, while progressively but efficiently increasing animat complexity only as necessary' [Wilson, 1991, 20ff.]. Both approaches begin with minimal ad hoc machinery and work upward synthetically from there, building larger complex machinery in order to support life and intelligence. Rather than focusing on the production of intelligent solutions, these approaches focus on the production of intelligent behavior, and the force driving the changes in behavior is evolution.

In AL, adaptation can be viewed as a kind of 'learning' that takes place on time scales much longer than the lifetimes of individual organisms' [Langton, *et al.*, 1992, 13ff.]. This class of learning is a product of the AL model: a population of organisms introducing changes by constructing (reproducing) other organisms, and natural selection providing the feedback as to how good, or how bad, the changes were by allowing only the strongest organisms to reproduce. The result is that over many generations, organisms evolve that are better suited for their environment. Since organisms normally mate and reproduce with organisms of the same species [Collins and Jefferson, 1992], this class of 'species learning' is always at the population or species level.

In contrast to evolutionary learning, 'individual learning' can be viewed as the learning that an organism does in its lifetime. AL, when considering individual learning, has combined the traditional methods of learning studied by the ML community, such as back-

propagation, with evolutionary learning. It has been generally found that a combination of species and individual learning works best. These models, however, examine only the effects of individual learning on the population, and do not examine how individual learning could evolve and function in an organism!

Unlike the AL approach, the Animats approach is concerned with the development of individual learning. However, with the notable exception of [Sutton, 1991], most of the Animats learning architectures are effective at trial-and-error learning and no more. What is missing from these architectures is the ability to perform planning or reasoning.

The approaches of GAs, CSs, AL, and Animats all take adaptation seriously. However, GAs and CSs usually adapt only structures of fixed length bitstrings (for two exceptions, see [Koza, 1989; Oppacher and Deugo, 1991]). These structures make for easy application of genetic operators, but their expressive power is insufficient for typical symbolic problem solving tasks required of many expert systems today. It is, to begin with, difficult to encode multi-valued features, n -place predicates ($n \geq 2$), and other 'higher-level' symbolic structures that contribute to a problem's description into fixed length strings of 0's and 1's. Strings of 0's and 1's also fail the *comprehensibility principle* [Michalski and Kodratoff, 1990]: knowledge created by a system should be easy for humans to interpret and comprehend, and should correspond to the way it is used and produced. Strings of 0's and 1's are neither easy to interpret, to comprehend, nor to construct. An alphabet of two symbols restricts the encoding for any one 'gene', the basic component of a 'chromosome string', to only two possibilities. Knowledge in symbolic AI systems is often couched in a rich symbolic notation, for example, frames. A frame can represent many more values than a single binary gene.

In summary, the approaches to intelligence mentioned above have the following advantages and disadvantages:

Approach	Advantages	Disadvantages
Symbolic AI	- has shown good results in narrowly defined domains	- is brittle - has shown only isolated competences
CBR	- does not work from scratch - adapts cases to match existing problem	- uses hand-coded adaptation rules - never adapts its adaptation rules - lacks a facility to opportunistically look for better solutions
Subsymbolic AI	- is robust - is fault tolerant - handles contradictions	- is weak in domains requiring high-level knowledge processing
GAs & CSs	- are fast - take adaptation seriously - benefit from previously acquired knowledge	- lack transparency - often use fixed size binary structures
Neural Networks	- are robust - are fault tolerant - handle contradictions	- often use hand-coded, fixed connections - lack transparency
Artificial Life	- focuses on behavior not solutions - makes evolution and adaptation essential components of model	- only considers species level learning
Animats	- focus on behavior not solutions - make evolution and adaptation essential components of model - consider individual learning	- are effective at only trial-and-error learning

Table 1.1 Summary of Different Approaches to Intelligence

1.2 CompEE's Approach to Intelligence

In this thesis, we present a new approach to intelligence that overcomes many of the disadvantages of the approaches mentioned in the last section by combining their strong points and using evolution as the main source of information gain.

Research into CompEE explores how individual learning could evolve and operate in an organism within a simulation model. CompEE takes a population of simulated organisms (SimOrgs), places them in an environment and has them learn useful features of the environment and behaviors that can be characterized as sequential decision problems [Grefenstette, 1992]. Unlike Grefenstette's Samuel system, whose only purpose is to learn a global, single set of decision strategies, represented as simple, fixed size condition action rules, CompEE describes the structure and growth of a SimOrg's knowledge, enabling it to survive and prosper in its environment, at many different operational levels in terms of biology and evolution. In CompEE's approach, a SimOrg does not use fixed size, simple rules; rather, it uses many different structures to encode information about its environment that are appropriate for each of its three levels: the genotypic level, the phenotypic level, and the cognitive level. For example, it decides what features are useful for building reliable simple and complex schemata¹ at the cognitive level; it decides what hierarchical organizations are best suited for its motivational drive hierarchy and for its chromosome hierarchy at the genotypic level; and it decides which gene values enable it to survive in the environment at the phenotypic level.

Each of CompEE's levels has an influence on the others that diminishes the further apart they are from one another. Knowledge that must be obtained immediately - possibly in order to keep a SimOrg alive - must be learned at its cognitive level. New physical structures that enable it to perform better in the environment must be determined at the

¹ Schemata are both declarative and procedural knowledge about the environment.

phenotypic level. The description of the physical structures must be determined at the genotypic level. Each level acquires information that is important for the others.

At the cognitive level, symbolic knowledge, in the form of a schema, is produced by a new reasoning method based on the process of evolution. This new method of reasoning is called Evolution-Based Reasoning. Similar to CBR, EBR attempts to reuse past knowledge. However, in contrast to CBR, EBR takes learning and adaptation to be integral components of the reasoning method. The construction of knowledge is construed as a process of differential selection on variation, similar to the one that governs the evolution of SimOrgs.

The phenotypic level is constructed from four simple systems: a Motivational System, a Stimulus System, a Memory System, and a Problem-Solving System. These systems enable a SimOrg to obtain regularities about the world, to detect properties about the world that can directly affect its characteristics and ultimately its genome, and to learn which observed changes can be ignored in general or for certain purposes (not all detectable properties are relevant).

The Motivational System represents a SimOrg's internal state and helps it to select a goal to pursue.

The Stimulus System has control over a SimOrg's five senses of sight, smell, touch, taste, and hearing. Its responsibility is to return a set of stimuli conjunctions that can be detected from objects at the SimOrg's current position in the environment.

The Memory System controls a SimOrg's two types of memories: long-term memory and short-term memory. The purpose of short-term memory is to store recently perceived information. Long-term memory consists of a finite stream of schemata. A schema is a tripartite structure, consisting of a conditional context, an action, and an expected context, and is used to represent both episodic and semantic information, i.e. information about specific events or episodes which occur in a particular place at a particular time, and de-

contextualized facts about the entities and relations between entities in the world [Eysenck and Keane, 1990].

The Problem-Solving System has the responsibility of determining how to meet the SimOrg's current needs and links the cognitive and phenotypic levels. Using the schemata in long-term memory and the current context (the current perceived external environment, and the active drive), the Problem-Solving System attempts to locate - using EBR - a schema whose action, once performed, will decrease the SimOrg's active drive.

CompEE's most primitive level is called the genotypic level. It is at this level that a SimOrg's description is stored - in a single genome - and adapted. A SimOrg's genome contains its physical description (parameters), e.g. its memory capacity, its life span, the sensitivity of its sensors, its control parameters such as mutation and crossover rates; and the organization of its motivational system, e.g. the precedents of its drives. The genotypic level's responsibility is to learn the correct values and organization of the parameters that will enable a SimOrg, and its future offspring, to survive in the environment and produce stable offspring.

Like most traditional AI approaches to intelligence, CompEE uses symbolic structures such as hierarchies. However, unlike those approaches which often use hand-coded rules to alter their structures, CompEE, as in the subsymbolic approaches of GAs and CSs, uses the process of evolution to manipulate its structures. Therefore, unlike CBR, CompEE makes evolution and adaptation essential components of the model - they are not treated as just welcome side effects but become the primary goals. For example, the construction of knowledge is seen as the process of differential selection on variation. Knowledge evolves in a manner similar to the evolution of a SimOrg: the survival of the fittest. Also, like AL and Animats, CompEE is concerned with the evolution of cognitive mechanisms. Rather than only simulating how a mechanism such as condition dropping may operate, CompEE attempts to understand how and why it evolved. In CompEE, evolution is not only used to

produce new structures, such as those in support of cognitive phenomena, but it is described as a cognitive process.

CompEE acquires information in structures located at different levels. The information at one level helps determine what information can be discovered at the next. Unlike GAs, AL, and most Animat models, CompEE is ultimately concerned with how an individual comes to gain reliable information about its environment in its lifetime, not how the species gains this information over several generations.

In summary, CompEE compares and contrasts with the other approaches to intelligence mentioned in the previous section in the following ways:

Similar To	Different From	Reason
Symbolic AI, AL, CBR, Animats	GAs, CSs, NNs	- uses symbolic structures
AL, Animats	Symbolic AI	- concerned with the evolution of cognitive mechanisms
GAs, CSs, AL, Animats	CBR	- makes evolution and adaptation essential components of the model
GAs, CSs, AL, Animats	CBR, Symbolic AI	- views evolution as the main process of knowledge growth
CBR, NNs, CSs, GAs	AL, Animats	- uses background knowledge
Symbolic AI	Animats, AL, GAs	- concerned with individual learning

EE	Existing Cognitive Models	<ul style="list-style-type: none"> - shows that learning occurs at many different levels - uses a hierarchy as the primary evolutionary structure - does not preprogram any control parameters or knowledge
----	------------------------------	--

Table 1.2 Comparison of CompEE with other Approaches to Intelligence

In this section, we have given some brief insights into CompEE's approach. The aim has been to sketch some of the relevant aspects of an evolutionary approach to human cognition. The full details of CompEE are given in the remaining chapters.

1.3 EE Predecessors

Evolutionary Epistemology, which inspired some of the ideas for CompEE, is almost as old as evolutionary thinking itself. Its current understanding has been developed from contributions of biologists, psychologists, philosophers, and neurobiologists.

The first version of EE was formulated by Herbert Spencer in his magnum opus, 'The Synthetic Philosophy'. In that work he argued for the idea of progress in evolution and showed the links between organic evolution and the development of society and culture. Without ever using the term EE, Darwin's 'The Origin of Species' [Darwin, 1859] gave strong empirical support to Spencer's version of it. It is curious that Darwin said in his Autobiography [Darwin, 1969, p. 109] he did not profit from Spencer's writings, even though he, too, believed that animal and human behavior could be understood only within the context of evolution. Darwin's insight and efforts led the way to the foundation of

modern ethology, and biologists such as Thomas H. Huxley and Ernst Haeckel took his ideas and argued further that all human intellectual capacities depend on the evolution of organic structures.

Konrad Lorenz explicitly formulated one of the central theses of EE, the idea that the Kantian *a priori* is to be regarded as an 'evolutionary *a posteriori*'. For him, the central nervous apparatus is adapted to the real world, 'just as the hoof of the horse is adapted to the ground of the steppe which it copes with' [Lorenz, 1941]. Based on a large body of observation in ethology, Lorenz replaced the doctrine of the *tabula rasa* with the thesis that an organism is equipped with innate dispositions that are the outcomes of evolution.

At the same time as Lorenz, philosophers such as Popper [Popper 1972], Campbell [Campbell 1976b], Hull [Hull, 1981], and Dawkins [Dawkins, 1976] developed evolutionary interpretations of the growth of knowledge. All would agree with the argument that 'the cognitive abilities in animals and humans are results of organic evolution and, consequently, all human knowledge capacities are products of natural processes' [Wuketits, 1990]. This amounts to a naturalized epistemology that is supported by results from biology, psychology, and neurobiology.

Jean Piaget provided some of the psychological evidence for EE from his studies of the development of cognitive capacities in children. His genetic epistemology [Piaget, 1970] attempts to explain the human development of conceptual thought, perception, and representation of the external world as an evolutionary dynamically organized process. The main contribution of this work to EE is the demonstration of the links between psychology, genetics, and epistemology.

Gerald Edelman's work on his theory of neuronal group selection provides additional neurobiological evidence for EE. The theory of neuronal group selection applies population thinking to the nervous system: selection operates during the lifetime of an individual to enhance the activity of parts of the nervous system that have a competitive advantage over

other parts and that produce responses with adaptive value for the individual. In his view, the brain is a selective system [Edelman, 1987].

The basic concepts of EE can be traced back to the evolutionists in the nineteenth century: Spencer initiated the argument and Darwin provided the evidence for it. In the twentieth century many philosophers adopted an evolutionary approach to an understanding of knowledge. Further evidence for EE has come from different scientific disciplines, such as biology, psychology and neurobiology. The realization that knowledge is generated by mechanisms older than any rational operation is hard to ignore.

1.4 Organization

Before getting to the specific details of the thesis, chapter 2 further outlines the philosophical background material from which the idea for CompEE emerged. Since CompEE is based on the philosophical view of Evolutionary Epistemology - a view that uses the process of evolution repeatedly - the chapter's first section provides a short introduction to the process of evolution and its terminology. There are three interconnected, yet distinct, versions of EE. The next section describes these versions and establishes the one that CompEE is derived from. The following section discusses some objections to EE and responds to them. The next section draws some conclusions about EE, and the final section summarizes the chapter.

Chapter 3 begins the detailed discussion on CompEE and outlines its major components. Since this thesis uses different notions of learning, in order to avoid confusion when discussing them, the first section identifies three different classes of learning, based on changes to the internal organization of a phenotype. In the following section, CompEE's computational approach, its fundamental tenets, and its accomplishments are discussed. Further sections describe CompEE's SimOrg and its three interconnected levels. The final section summarizes the chapter.

The next three chapters discuss each of the three levels in detail. In chapter 4, we discuss the genotypic level. Every SimOrg's genotypic level contains a single genome. The first section describes the motivation for our choice of hierarchies to represent the genome's two components: the chromosome hierarchy and the motivational drive hierarchy. After describing each of these components, the following section describes how the genome and its components are adapted using the process of evolution, and how their evolution results in species learning. The final section summarizes the chapter.

In chapter 5, we discuss the phenotypic level. The chapter's first four sections describe the four systems that comprise the phenotypic level: the Motivational System, the Stimulus System, the Memory System, and the Problem-Solving System. Of particular importance is how schemata are used to represent knowledge by the Memory System and the Problem Solving System, how they are generated for the Problem Solving System, and the internal and external views of a schema. The next two sections discuss how a SimOrg evolves and how its four systems are used to control its behavior in the environment. The final section summarizes the chapter.

In chapter 6, we discuss the cognitive level. The chapter's first section describes how reliable, useful symbolic knowledge, in the form of a schema, is produced using a new model of reasoning called Evolution-Based Reasoning. Since EBR requires each schema to have a fitness value, the following section discusses how this value is calculated. EBR also requires genetic operators for altering schemata. The next section describes four such operators used by EBR: mutation, crossover, split, and join. The final section summarizes the chapter.

In chapter 7, we discuss some of our findings generated from the Smalltalk implementation of CompEE. We examine the model's behavior in different environments to demonstrate that SimOrgs are indeed evolving and learning reliable information about the environment as expected.

The concluding chapter summarizes and discusses several open problems and extensions that were noted during the research.

1.5 Chapter Summary

Although traditional AI has proposed many different models of specific human abilities, many of them will work only under very specific conditions, e.g. there is a large amount of background knowledge to work from, there is no conflicting information or conflicting goals in the environment, or their parameters can be hand-set to enable them to work correctly in the current environment, and many of them lack facilities to opportunistically look for better solutions and manage novel arrangements. However, in many situations, a complete set of background knowledge is impossible to determine, conflicting goals can not be prevented, and correct parameter settings are unknown. In these situations, the ability to produce novel solutions and methods of determining whether the novel arrangements are useful is required.

In unpredictable environments, where an organism's success relies on its creativity, Riedl [Riedl, 1984] has shown that we, and even lower level organisms, gain information by a mechanism of trial and error and of experience and expectation. He also showed that certain expectations are preprogrammed by experience, e.g. our expectation to believe as true what has often been confirmed.

This cycle of experience, adaptation and expectation can be described as the process of evolution operating at many different levels in an individual. Our innate expectations are based on mechanisms that have resulted from the experiences, adaptations and expectations made by individuals over many generations, and our currently held expectations are based on the same cycle of events made in our recent past. CompEE is the computational approach to this view of human cognition.

Chapter 2

The Origin and Evolution of Evolutionary Thought

The study into the nature and ground of experience, belief, and knowledge is known as Epistemology. Knowledge forms the main topic of Epistemology, along with other cognitive notions such as understanding, reason, judgment, sensation, imagination, supposing, guessing, learning, and forgetting. Questions about knowledge include its nature, its types, its origin, and what is known. Epistemology attempts to explain what we can know, and how we know anything.

There have been several different views in philosophy as to the source of knowledge, such as Rationalism (Intellectualism), Empiricism, Transcendental Idealism, Logical Positivism, and Evolutionary Epistemology (EE). The view of EE is that knowledge development is evolution carried on by another means.

'Knowledge development is a direct extension of evolutionary development, and the dynamics of the two processes are identical' [Hahlweg & Hooker, 1989b].

Wuketits' [Wuketits, 1984b] has defined five postulates for EE:

- 1) All organisms are equipped with a system of innate dispositions; no individual living system is initially a 'clean slate' or tabula rasa.
- 2) Innate dispositions are the outcome of natural selection; they are the products of selective mechanisms, which, among all 'initial products', favor and stabilize the one which best copes with the conditions of living and surviving.
- 3) All psychic phenomena in the subhuman world as well as mental abilities proper to human systems (self-consciousness) are based on biological structures and functions; biological evolution has been the precondition to psychological and mental evolution².
- 4) Nature is objective; it has existed before and independently of an observing subject. If nature was not real, it could never be observed.
- 5) Evolutionary Epistemology is an interdisciplinary approach to explaining and understanding epistemic activities; it is based on biological and psychological research and corresponds with results in the fields of linguistics, anthropology, ethology, [computer science]³, and sociology.

These postulates are representative of three interconnected, yet distinct, versions of EE. The first version attempts to account for the evolution of the structures and processes of cognition in animals and humans by extending the theory of evolution to those aspects of animals which are the biological substrates of cognitive activity [Lorenz, 1973], and is often referred to as the EE of cognitive mechanisms (EEM) [Bradic, 1986]. The second

² Psychological phenomena are common to all organisms who share similar structures, such as a nervous systems; Mental phenomena are at a higher level and are due to specific brain activities.

³ My addition.

version attempts to account for the evolution of scientific ideas and culture using models and metaphors from evolutionary biology [Popper, 1974; Dawkins, 1982], and is often referred to as the EE of theories (EET) [Bradie, 1986].

The third, rarely mentioned, version of EE, but of primary concern to this thesis, attempts to account for the evolution of animal or human knowledge throughout their lifetimes by combining EEM with EET applied to an individual's knowledge, not to scientific ideas and culture. This version of EE links the views of EET and EEM, and we shall refer to it as the EE of individual knowledge (EEI). As Lumsden and Wilson [Lumsden and Wilson, 1981] have pointed out, the link between the evolution of cognitive mechanisms and the evolution of culture and science has been little explored, and the justification of EE is to find the missing link.

The three versions of EE can be considered as Naturalized Epistemologies [Quine, 1971]. They differ from more traditional approaches to epistemology because they are primarily descriptive rather than normative approaches. Instead of addressing the traditional issues of epistemology, such as what we can know, how we know it, and how we distinguish true belief from knowledge, these approaches offer accounts of how we come to know and come to have the kinds of knowledge we have. In addition, while a traditional epistemology does not use scientific evidence and theories to answer epistemological questions, EE does.

For the remainder of this chapter, we examine in detail the different versions of EE, from which the idea of CompEE emerged, and discuss some objections to EE. Since these discussions require an elementary understanding of the process of evolution, the following section is provided to acquaint the reader with it.

2.1 Evolution

What is evolution? Literally, evolution means: an unfolding; a process of development or change; or a movement that is part of a series [Guralnik, 1983]. In biology, evolution is generally defined as 'a change in the gene frequency of a population' [Arthur, 1984; Wilson and Bossert, 1971], 'the process by which some genes become more numerous and others less numerous in the gene pool' [Dawkins, 1976], or 'the natural unfolding and change of organisms down through the generations, from earlier forms, widely different' [Ruse, 1986].

What are the objects of evolution? From the point of view of molecular evolution [Ayala, 1976b], it is the genes in a population of organisms that arise and change over time. From the point of view of morphological evolution [Arthur, 1984], it is the distributions of organisms that change over time. From the point of view of Evolutionary Epistemology [Callebaut and Pinxten, 1987; Campbell, 1987b], it is an organism's knowledge, thoughts, concepts, strategies and behaviors that change over time. Generalizing these views, we abstractly define evolution as the change in unique collections of entities (e.g. genes, organisms or knowledge) where such entities are drawn from shared pools. We have not, however, said anything about how the change occurs?

Darwin's *Origin of Species* [Darwin, 1859] established the fact of evolution - that organisms have descended with modifications from common ancestors - and provided a theory to explain it. Darwin's theory proposed that different species of plants and animals have arisen by a process of slow and gradual changes over successive generations, brought about by Natural Selection [Hale & Margham, 1988]. The main points of his theory are:

1. Sexual reproduction produces a wide range of variability.
2. All living forms have the potential for a geometric rise in numbers.

3. There must be a struggle for existence in which well suited individuals to the particular operating conditions are more successful at breeding than others because populations usually remain within limited sizes.
4. The struggle for existence results in natural selection that favors the survival of the best-adapted individuals.

The current view of evolution theory, combining Darwin's theory of evolution and Mendelian genetics [Mendel, 1981], is referred to as neo-Darwinism or the modern synthesis. Mendel discovered the existence of discrete hereditary elements (genes) and the principles of their inheritance by offspring from parents. Mendelian principles of inheritance resolved the one weakness in Darwin's theory: ignorance of the nature of heredity.

How does evolution occur? Evolution by Natural Selection involves the cooperation of three processes: morphogenesis, natural selection, and reproduction. Morphogenesis is the process by which a fertilized egg develops into a mature organism. A organism is initially constructed by translating the genetic information (genes), contained in its genome, into phenotypic structures. The interaction of an organism with its environment, using its current implementation, results in its further development. The concept that bodily complexity emerges by a developmental process of interactions between the genes and their environment from a zygote, rather than being totally mapped out in the egg, is known as epigenesis. Generally, to quote the abridgment of Weiss' comments by Løvtrup:

'Epigenesis ... presumes that developing systems start from a rather primitive, homogeneous, chaotic, at least, lowly organized condition into which increasing complication and real ... diversification come in progressively as development proceeds;... the later parts of the organism are

not as such pre-existing in the germ but are gradually established by a process of individualization.' [Løvtrup, 1974]

Natural selection is the process that identifies those organisms that are well suited for their environment - the ones that are alive, fit, and healthy. With constraints such as limited food and space, organisms face a constant struggle for existence. This struggle for existence results in only the best-adapted organisms surviving. Organisms are better suited for their environment if they have an advantage over other organisms in the same environment. The advantage may arise due to a change in an organism's physical structure, enabling it to survive better in its current environment; it may be due to a change in the environment, making an organism stronger than others in the new environment; or it may be due to a combination of changes in an organism and its environment.

Reproduction is the process in which two parents produce offspring whose genetic constitution is a mixture of two, potentially different, gametes, resulting in the inheritance of parental traits⁴ by the offspring. It is important that fit parents produce more than one offspring in their lifetime. The reason is simple. If fit parents produce only one offspring, they replace only one of themselves in the next generation, which will eventually lead to a loss of their fit genes to the population - assuming their offspring have the same reproductive rate.

These processes do not ensure by themselves that evolution will occur; four conditions must also be met:

⁴ In fact, it is not the traits that are passed to one's offspring, but rather the specification of the traits. For example, from the point of view of molecular evolution, it is the genes that are passed to one's offspring.

- 1) *Variation*: There must be variation in the traits of the organisms in the population. The variation may occur due to morphogenesis - organisms with similar genes may develop differently. However, variation in the offspring is more likely to occur during reproduction, when new combinations of genes form new traits, and when gene mutations alter the traits of the parents. Without variation, organisms will not be able to change with their environments, and without change, the species will eventually become extinct.
- 2) *Inheritance*. There must be some way of passing on the desirable traits or their specification to the offspring. Without inheritance, successful traits will never be incorporated into any organism - organisms would be nothing more than random gene compositions. If desirable traits can not be maintained by the species, it will not be able to adapt to its environment and will eventually become extinct.
- 3) *Differential reproductive success*. Fit organisms must leave greater numbers of offspring, in succeeding generations, than weak organisms. The fitness of an organism is determined by its interaction with the environment - fit organisms survive to reproduce. Without differential reproductive success, no increase in the number of organisms would be seen with traits that lead to the selective advantage. Without the increase in numbers, the species will not be able maintain the successful traits and adapt to its environments.
- 4) *Constancy*. There must be some degree of environmental constancy through time for the selective advantages of traits to be incorporated into the species.

Without constancy, new traits will never have time to demonstrate their selective advantage to the population.

There is little doubt about the fact of evolution, about Darwin's theory of evolution, or about Mendelian Genetics. However, by using our description of evolution, neo-Darwinism need not only work with genes. As described, evolution can work with any population of entities - genes, organisms, or knowledge - provided that the four conditions for it are met and the processes of morphogenesis, natural selection, and reproduction are active.

2.2 Evolutionary Epistemology Programs

Evolutionary epistemology is a theory of knowledge development analogous to evolutionary theory. EE takes seriously the fact that evolution can be seen as a knowledge gaining process, and that since knowledge is a product of our biology, and since our biology is a product of evolution, knowledge can be explained in terms of evolution.

In general, EE applies the mechanism of differential selection on variation - evolution - to the growth of knowledge. 'In this sense knowledge is seen to be a function of success in a field of alternative ideas' [Thompson, 1988]. EE maintains that this mechanism can be generalized to other activities such as learning and science [Plotkin, 1986]. While some proposals to learning and science simply evoke randomness as a source of novelty, as may be the case with Thorndike's 'method of trial, error, and accidental success' [Thorndike, 1987], and as seems the case with Monte Carlo simulation and the Artificial Intelligence (AI) paradigm of 'generate and test', proposals of EE are based on a clear understanding that randomization alone is not powerful enough. Only repeated rounds of random variation followed by non random selection suffice to produce complex new structures. These versions of EE all have the following in common: they believe that the cognitive abilities in

animals and humans are results of organic evolution, and, consequently, all human knowledge processes are products of natural processes [Wuketits, 1990]; and they attempt to explain the cognitive capacities of humans in terms of biological evolution. EE is not a small project. For a recent extensive (600 items) bibliography of EE, see Campbell [Campbell *et al.*, 1987].

The implications of an evolutionary theory for epistemology have been studied by many: [Piaget, 1970; Toulmin, 1972; Lorenz, 1973; Popper, 1987; Campbell, 1987b]. Typically, these authors have tried to account for scientific knowledge growth by appealing to a mechanism of differential selection on variation, operating in a field of competing theories. Theories with higher confirmation, or greater explanatory power (or other, often pragmatic, features indicative of 'epistemic success', 'strength' or 'fitness') win the struggle for incorporation into the body of accepted scientific knowledge.

There are essentially three strands of EE: EEM, EET, and EEI, all of which start with the fact of evolution. The first version, EEM, studies *the evolution of cognitive mechanisms, structures and processes - such as nervous systems, brains, and sensory organs - that support cognitive capacities*, with the aim to learn something about the limits of cognition from an understanding of these underlying biological mechanisms and their development. This form of EE is a biological theory; cognition is treated as a biological phenomenon by extending biological evolution to those traits of animals and humans that are the biological substrates of cognition - *the literal extension of biological evolution*.

Lorenz was one of the first biologists to attempt to find insights into epistemological issues through biological research. His research, often referred to as Bioepistemology [Hahlweg & Hooker, 1989b], is concerned with cognogenesis - the evolution of the structures and process of cognition. Indeed, Lorenz [Lorenz, 1973] holds that life itself is a cognitive process because the structures of living things encode properties of the world they live in. This Austro-German school of EE views the Kantian categories of thought

and the categorical forms of intuition [Kant, 1781] as adaptations, as parts of our hardware, and certainly not beyond criticism. Lorenz wrote that these categories should be understood in terms of evolution and that they are for us 'inherited working hypotheses'.

'Our working hypothesis should read as follows: Everything is a working hypothesis. This holds true not only for the natural laws which we gain through individual abstraction a posteriori from the facts of our experience, but also for the laws of pure reason. The faculty of understanding does not in itself constitute an explanation of phenomena, but the fact that it projects phenomena is due to its formulation of working hypotheses, developed in evolution, and tested through millions of years.' ([Lorenz, 1941] pp. 132)

From Lorenz's point of view, any a priori is an a posteriori of evolution [Wuketits, 1990], meaning that what cognitive structures and processes animals and humans use to perceive and represent the world are the accumulated experience of their predecessors.

There is one major objection to this version of EE. Although it may explain why commonsense works - as far as it does - it does not have anything to say about knowledge (or about anything beyond the realm of common sense), and nothing at all about the growth of knowledge. Therefore, it is not really an epistemology. However, it does tell us about the foundations of experience and knowledge development, and is, therefore, a descriptive approach to epistemology.

Another of the biologically based, scientifically oriented approaches to epistemology, called genetic epistemology, was developed by Jean Piaget. In his own words:

'Genetic epistemology attempts to explain knowledge, and in particular scientific knowledge, on the basis of its history, its sociogenesis, and

especially the psychological origins of the notions and operations upon which it is based. These notions and operations are drawn in large part from common sense, so that their origins can shed light on their significance as knowledge of somewhat higher level.' ([Piaget, 1970] pp. 1)

Being a psychologist, Piaget's genetic epistemology was an investigation of the psychology of knowing. He outlined six stages, each identified by a new elaboration of sensorimotor activity, for the logical and rational organization of knowledge based on corresponding formative psychological processes, and he was mainly concerned with psychogenesis: the cognitive ontogenesis of the individual. Although Lorenz was looking at biological processes and Piaget was looking at psychological processes, both would agree that through evolution the structures, qualities and dynamics of the world are encoded into an organism.

The second version of EE, EET, studies the development of culture - *the growth of science and other forms of rational knowledge* - using models analogous to evolutionary biology. This form of EE sometimes takes the form of a methodology or metatheory [Popper, 1974], or a sociobiological theory [Dawkins, 1982]. This form of EE *is an analogous, metaphorical extension of biological evolution*. For example, in Popper's EE:

- 1) Theories are never due to induction; they far outstrip sensory experience and are invented, not derived from experience.
- 2) Theories are never verifiable but only falsifiable, and because of this, only the fittest theories survive the severe tests.

The analogy with Darwin is: truth is like adaptation. Just as no organism is perfectly adapted, no theory can ever be known to be perfectly true; just as mutations are not 'induced' but random, theories are not 'induced' from experience but are freely invented by our minds; in the end, there are just built-in physiological 'theories' and innate drives: to explore, to correlate, and to imitate (see reply to Campbell in [Schilpp, 1974]). There is progress towards the truth as we consciously apply the method of trial-and-error. Popper also agrees with Lorenz and Piaget that in the present we benefit from the evolution of our species. He believes that there are some kinds of theories or theoretical preconceptions that guide our thoughts and actions [Popper, 1972].

As Popper realized, science has an anti-evolutionary characteristic - its aim is to increase the unification of knowledge - while the emphasis in the path of organic evolution is on fanning out. Popper's EE, therefore, exhibits only a very weak, merely formal analogous extension of biological evolution to the theory of the growth of knowledge⁵.

Like Popper, Dawkins believes that 'for an understanding of the evolution of modern man, we must begin by throwing out the gene as the sole basis for our ideas on evolution... The gene will enter my thesis as an analogy, nothing more' [Dawkins, 1976, pp. 205]. Instead of using a gene, Dawkins prefers the notion of replicator. A replicator is anything in the universe which makes copies of itself. A gene is certainly a replicator, and so is knowledge. Dawkins call his knowledge replicator a meme. Examples of memes include: tunes, ideas, catch-phrases, clothes, and so on. A meme is anything that can 'reproduce' itself in the meme pool by imitating from person to person; for example, the passing on of a good idea. Like Popper, this weak, analogous use of evolution also has a lack of commitment to an evolutionary understanding of epistemology. It may say

⁵ Scientific knowledge.

something about how knowledge is transmitted, but says nothing about its nature, origin or what is known.

Campbell's version of EE [Campbell, 1987b] combines the EEM and EET versions. In particular, he approves of Lorenz's biologizing of Kant's notion of categories - an EEM point. He believes that these categories, of perception and of thought, have evolutionary origins. Campbell recognizes that 'an evolutionary epistemology would be at a minimum an epistemology taking cognizance of and compatible with man's status as a product of biological and social evolution' [Campbell, 1987b]. He also maintains that his 'blind-variation-and-selective-retention' model explains not only the evolution of biological structures, but also the growth of scientific knowledge - an EET point. He argues that evolution is a knowledge process and that the paradigm of natural selection can be generalized to other activities, such as learning, thought, and science.

Campbell's blind-variation-and-selective-retention model advocates the following:

- 1) A blind-variation-and-selective-retention process is fundamental to all inductive achievements, to all genuine increases in knowledge, and to all increases in fit of a system to the environment.
- 2) In such a process, there are mechanisms for variation and preservation and/or propagation of the selected variations, and a consistent selection process.
- 3) Processes which shortcut the blind-variation-and-selective-retention process are themselves produced by a blind-variation-and-selective-retention process.
- 4) These shortcut processes contain a blind-variation-and-selective-retention process at some level of their operation.

According to Campbell, when you have a problem, you consider a number of chance ideas in your mind, vary them, throw away the bad ones, retain the best ones, and repeat until a good idea is found. This may sound overly simple, and one might argue that today's discoveries are not made in this random fashion. However, Campbell would not argue this point. He would say that the guiding principles that are now used were themselves created by the same process.

Campbell, by expanding and applying his model to a biological and social evolution setting, has described the human knowledge process in terms of numerous, hierarchically related mechanisms, with some form of blind-variation-and-selective-retention process active at each level of the hierarchy. He does not, as Popper does, consider only the level of scientific knowledge. He has identified ten discrete levels acted upon by his blind-variation-and-selective-retention process: non-mnemonic problem solving, vicarious locomotor devices, habit, instinct, visually supported thought, mnemonically supported thought, socially vicarious exploration: observational learning and imitation, language, cultural cumulation, and science. Once a level is accepted by natural selection, because of its selective advantage to the genotype, it is considered 'knowledge', and is in turn used to create more complex knowledge. As Campbell says: 'What are criteria at one level are but "trials" of the criteria of the next higher, more fundamental, more encompassing, less frequently invoked level' [Campbell, 1987b].

Where Popper started and stopped at the level of scientific knowledge, Campbell's EE is a theory of human knowledge that attempts to understand it as continuous with animal knowledge, and to understand its discontinuity, if any!

Other than Campbell, the third version of EE, named EEI by this thesis, is rarely discussed at length [Bradie, 1986; Wuketits, 1990 (Ch. 3)], and often only as a subpart of EET. EET is concerned with, for the most part, the historical evolution of science - the phylogenetic development of knowledge by humans. EEI is concerned with the evolution

of an individual's knowledge in its lifetime - the ontogenetic development of knowledge by an individual. Toulmin characterizes the distinction between EEI and EET as follows:

'We ... face questions about the social, cultural, and intellectual changes that are responsible for the historical evolution of our various modes of life and thought - our institutions, our concepts, and our other practical procedures. (These questions correspond to questions about phylogeny in evolutionary biology.) Individually speaking, we ... face questions about the manner in which maturation and experience, socialization and enculturation shape the young child's capacities for rational thought and action - how the child comes to participate in his native society and culture. (These questions correspond to the questions about ontogeny in developmental biology.)' ([Toulmin, 1981] pp. 26)

Thus, EEI forms the link between EEM and EET. EEM examines the development of the biological structures necessary for individual knowledge; EEI, using the developed structures, examines the development of an individual's knowledge, which is the source of scientific knowledge; and using every individual's knowledge, EET examines the development of the scientific knowledge of society. Schematically:

Figure 2.1 Evolutionary Epistemology Program Connections

One crucial link in this path, as Bradie points out, is the biologizing of Kant's categories:

'Seen from the perspective of the EET program, the categories are not only descriptive of how organisms cognize and evaluate, but prescriptive as well. Thus ... one would have a biological analysis and explanation of some normative features of the behavior of organisms.' ([Bradie, 1986] pp. 409)

Another crucial link in the path is the demonstration that the perceived sequential train of one's thoughts is the outcome of evolution, where pieces of thoughts are put together and compared with memories as to their appropriateness in the current situation, and the best 'fitting' one, i.e. the one with the best 'match', selected as the next part of the 'train of thought'. The existence of the EEI link would connect the EEM and EET approaches to EE, and provide empirical evidence for Campbell's blind-variation-and-selective-retention model.

2.3 Objections to Evolutionary Epistemology

Several people have argued that the analogy between the evolution of the species and the growth of knowledge is defective [Thagard, 1988; Ruse, 1986]. And even Campbell in his early position noted five problems [Campbell, 1987a]. In this section, we identify some common objections to EE and, when possible, provide the arguments against them.

The first objection is that evolution by natural selection is tautological. The argument begins with noting that one of EE's principle canons is the survival of the fittest. Next, the argument states that fitness can only be measured by survival. If one accepts this, the argument concludes that the canon asserts the survival of the survivor - a tautology. This

argument can be refuted easily by showing that fitness can be measured without resorting to survival. This is of course true, fitness can be measured by resorting to speed, quality, quantity, or stability. The point is not how well it can be measured, but that there are factors other than survival that can be used to represent fitness.

The second objection is that EE is circular. The argument begins by noting, as Wuketits' fourth postulate for EE points out, EE is connected with the position of hypothetical realism. Next, the argument states that the main tenants of hypothetical realism are that all knowledge is hypothetical, i.e. conjectural, fallible, uncertain; there exists a real world independent of our consciousness; and the real world is partially knowable by perception and experience. According to this position, all knowledge is hypothetical. This statement is itself part of a theory, and thus knowledge; therefore, it must apply to itself, and hence lead to a contradiction. The discussion showing that this objection is erroneous can be found in [Vollmer, 1987] and is much too long to consider here. It begins by establishing that hypothetical realism does not declare that all statements are hypothetical; it claims only that synthetic statements are hypothetical. Using this statement, it goes on to show that hypothetical realism is not self-refuting.

Another objection is that, while in biological evolution mutations are random - or, as Campbell states, blind - changes to scientific knowledge are directed. Changes in science happen because they are needed. They are motivated by the fact that there is usually a goal or a new scientific problem to meet or solve, and they are guided by earlier results. Mutations in biology do not attempt to meet a goal or solve a problem. This is a common position for changes in science, but, as Bradie points out [Bradie, 1986], we do not yet understand the psychological and sociological dynamics of discovery. Therefore, one can't say for sure that the way an individual changes his theories is not by blind variation or the result of a process created by a blind-variation-and-selective-retention process. In fact, because evolution permits only the survival of fittest individuals from the environment, the

environment ultimately influences what genes are present in subsequent populations, and as a result what type of mutations can occur. Therefore, the environment can 'direct' what is changed in the future.

Another objection is that organic evolution is not progressive, whereas scientific evolution is. Obviously, there is a general progression in science, but is there a general progression in organic evolution? The answer is yes. Animals become extinct, just as theories do. Organisms survive because they work better than others. The same holds for theories. In fact, one could not deny that the evolution of a molecule to man shows progress. It seems apparent that the notion of progress needs to be redefined.

Another objection is that there is a considerable amount of hybridization in science (blending of different, disparate subjects and theories), while organic evolution is more of a fanning out process. However, there are special cases in plants where it happens that two separate evolving species do come together [Grant, 1981], though it may require several variants of each species before this hybridization occurs.

One of the most interesting objections to EE, espoused by Kordig [Kordig, 1982], is that it is self-referentially inconsistent. The argument is as follows:

- 1) The theory of EE, call it T, has the property E of being evolutionary.
- 2) The theory T holds that all properties of all biological systems, natural and intellectual, evolve.
- 3) Therefore, the property E evolves.
- 4) But, If E evolves, it changes and becomes what it now is not.
- 5) Therefore, T must eventually possess property not(E);
- 6) And, at that point, T will not be evolutionary.
- 7) But (6) contradicts (1).
- 8) Therefore, T must be necessarily false.

These mechanisms are essential for the processes of evolution, and, when combined, form the foundations of EE.

2.4 Conclusion

It is quite clear that research into Epistemology has evolved. Initially, normative approaches to epistemology were proposed. Since they did not use scientific evidence to answer epistemological questions, they had to rely on other, and we say less stable, foundations such as 'man's existence' and 'instincts'. EE's descriptive approach, based on the fact of evolution, recenters the epistemological problem to the growth of knowledge and discovery. In EE's view, since knowledge is a product of our biology, and since our biology is a product of evolution, knowledge can be explained in terms of evolution.

EE takes the evolution of species, the evolution of individual beliefs and belief systems, and the evolution of science to be not just analogous processes, but to have literally the same underlying mechanisms. Since we - our biology - are a product of evolution, so is our knowledge; ways of thinking have developed because they made survival possible. Learning strategies and information processing methods (and even particular beliefs) are what they are (and continue to evolve) because of their selective advantages.

While research into EE has provided many theories about the growth of knowledge, it has failed to advance detailed, empirically testable ideas as to what the physical structures of knowledge are, and how they reproduce? What components are required for knowledge to evolve? How is fitness actually measured? What role does behavior play in evolution? Although EE has attempted to analyze these topics, it has failed at their synthesis.

The reason for this may be due to the fact that EE frequently concerns itself with the growth of scientific knowledge, which lies well beyond individual learning. Whether the synthesis of cultural learning is possible is left as an open question. However, individual

learning is obviously a requirement for cultural learning, and its synthesis should be considered first.

The problem for many in accepting EE is that the current arguments for it are philosophical in nature. This is a suitable format for a philosophical audience, however, for an audience of people who want to make use of the ideas, those in AI, Machine Learning and cognitive science, a computational approach demonstrating the viability of such ideas is required. This thesis attempts to do just that: a computational approach to EE!

2.5 Chapter Summary

The study into the nature and ground of experience, belief and knowledge is known as Epistemology. There have been several different views in philosophy as to the source of knowledge: Rationalism, Empiricism, Transcendental Idealism, Logical Positivism, and Evolutionary Epistemology. The view of Evolutionary Epistemology (EE) is that knowledge development is evolution carried on by another means. There are three interconnected, yet distinct, versions of EE: EEM, EET, and EEI. EEM attempts to account for the evolution of the structures and processes of cognition in animals and humans by extending the theory of evolution to those aspects of animals which are the biological substrates of cognitive activity. EET attempts to account for the evolution of scientific ideas and culture using models and metaphors from evolutionary biology. EEI is concerned with the ontogenetic development of knowledge by an individual. Several people have argued that the analogy between the evolution of the species and the growth of knowledge is defective. Their objections focus on the areas of variation, selection, progress, and hybridization. All of the objections are a result of failures in the analogy between the evolution of the species and the evolution of scientific knowledge. However, all of the EE approaches have the essential mechanisms for evolution, mechanisms for variation, inheritance, and selection - the foundations of EE. The problem for many in accepting EE is

that the current arguments for it are philosophical in nature. This thesis attempts to produce a computational approach to EF.

Chapter 3

Computational Evolutionary Epistemology

Our approach to Evolutionary Epistemology is called Computational Evolutionary Epistemology: a biologically motivated approach to cognitive modeling and machine learning that uses abstractions of known biological mechanisms and simplified Darwinian processes, and developed for computational study of the evolution of individual learning and knowledge on a computer.

CompEE follows the approaches of Artificial Life, Animats, Genetic Algorithms and Classifier Systems, to intelligence. AL complements the traditional biological sciences, concerned with the analysis of living organisms, by attempting to synthesize behaviors normally associated with natural living systems within computers and other artificial media [Langton, 1989; Langton *et al.*, 1992]. The Animat approach advocates maintaining the holism of the situation of real animals in real environments, while progressively but efficiently increasing animat complexity only as necessary [Meyer & Wilson, 1991; Wilson, 1991]. Genetic Algorithms simulate mechanisms of Darwinian evolution to solve a wide range of problems [Goldberg, 1989; Belew & Booker, 1991], such as the Traveling Salesman Problem [Deugo and Oppacher, 1992]. A Classifier System is a machine learning system that, by using a GA as its primary discovery mechanism, learns syntactically simple

rules for guiding its performance in an arbitrary environment [Goldberg, 1989; Grefenstette, 1988]. The common theme of these approaches, including CompEE, is to create adaptive algorithms and machine learning systems by borrowing analogies from evolutionary theory and other areas of biology. For example, CompEE uses the process of evolution for adapting its structures. However, unlike the sequential, binary structures that are used by GAs, CompEE applies the process of evolution to tree-like structures and to the symbolic structures of traditional AI systems, e.g. facts, rules, plans and analogies.

CompEE is ultimately interested in evolution that takes place on a time scale of milliseconds in an organism's brain. CompEE asks questions like the following: is the very process of thinking an evolutionary one, i.e. is the perceived sequential train of thoughts the outcome of Darwinian selection, where pieces of thoughts are put together and compared with memories as to their appropriateness in the current situation, and the best 'fitting' one selected as the next part of the 'train of thought'? Does the brain operate as an Evolutionary Machine?

In this chapter, we discuss CompEE in more detail. However, in order to avoid confusion when discussing learning in the thesis, the next section identifies three different classes of learning, based on changes to the internal organization of a phenotype. It is obvious that the learning an individual does in its lifetime is different from the learning that a species, as a whole, does over several generations. For example, an individual accumulates knowledge in its brain, while a species accumulates knowledge in the genes. Many Animat systems claim to learn, but don't clearly identify the type of learning. This omission makes the systems often difficult to compare and contrast.

In the following sections we briefly describe CompEE's computational approach, its fundamental tenets, its SimOrgs and its accomplishments. CompEE's multi-level approach takes a population of SimOrgs, places them in an environment, and has them learn - at the appropriate level - useful behaviors and features of the environment.

3.1 What is Learning?

In AL, “evolutionary adaptation can be viewed as a kind of ‘learning’ that takes place on time scales much longer than the lifetimes of individual organisms” [Langton, *et al.*, 1992, 13ff.]. This class of learning is a product of the AL model: a population of organisms introducing changes by constructing (reproducing) other organisms, and natural selection providing the feedback as to how good, or how bad, the changes are by allowing only the strongest organisms to reproduce. The result is that over many generations, organisms evolve that are better suited for their environment. Even though individual organisms change between and during generations, this form of ‘evolutionary species learning’ is concerned with only those changes that are useful to, and maintained by, the species.

In contrast to evolutionary learning, ‘individual learning’ can be viewed as the learning that an organism does in its lifetime. AL, when considering individual learning, has combined the traditional methods of learning studied by the connectionist community, such as back-propagation, with evolutionary learning. For example, [Ackley and Littman, 1992] found that a combination of evolutionary and individual learning (as implemented with neural networks and back-propagation) works well. Similarly, [Belew, *et al.*, 1992] found that this particular hybrid worked well because of the combined effects of the global searching performed by the population and the local searching performed by the individuals. These models, however, examine only the effects of individual learning on the species, and do not examine how individual learning could evolve and function in an organism!

In order to avoid confusion when discussing learning, we propose three different classes of learning, based on evolutionary changes in the internal organization of a phenotype or animat that improve its performance:

- 1) *Species Learning*: This occurs when there are changes in the phenotype's genes as a result of phylogenesis.
- 2) *Development Learning*: This occurs when there are changes in a phenotype's structures as a result of epigenesis.
- 3) *Individual Learning*: This occurs when there are changes in the phenotype's organ subsystems, like the brain, as a result of information-gaining processes.

AL, Animat and GA research has been mostly concerned with the first two classes of learning [Collins and Jefferson, 1992; Ackley and Littman, 1992], but very little research into the third class. The last two areas have been examined extensively by workers in the EE community. We find it amazing that in [Langton, 1989; Langton *et al.*, 1992; Belew & Booker, 1991; Schaffer, 1989; and Meyer & Wilson, 1991] there are no references to any work on EE. It is equally surprising that the reverse is also true: in [Campbell *et al.*, 1987], a recent extensive (600 items) bibliography of research in EE, no reference is made to AL, GAs, or Animats.

3.2 What is CompEE?

CompEE is a form of EE that treats cognition as a biological phenomenon by extending biological evolution to those traits of animals and humans that are the biological substrates of cognition. CompEE attempts to show that to study intelligence, one can not simply study the manifestations of intelligence. Rather, one should consider the entire process of the development of intelligence.

CompEE is a computational approach to EEI. As mentioned in chapter 2, EEI forms the link between EEM and EET, and it attempts to unify the different versions of EE. As with

the EEM version of EE, CompEE studies the evolution of cognitive mechanisms, structures and processes that support cognitive capacities. CompEE, for example, studies the evolution of long and short-term memories, motivational drive hierarchies, innate schema templates, and Evolution-Based Reasoning. As with the EET version of EE, CompEE is an analogous, metaphorical extension of biological evolution. We take the evolution of species, evolution of individual beliefs and belief systems, and evolution of science to have literally the same underlying mechanisms. Since we are a product of evolution, so is our knowledge; ways of thinking have developed because they made survival possible. Some of our learning strategies and information processing methods (and even particular beliefs) are what they are (and continue to evolve) because of their selective advantages. Similar to Campbell's version of EE, CompEE describes human knowledge processes at various hierarchically related levels, with a process of evolution at each level. However, unlike Campbell's EE version that has six levels to support individual learning, CompEE has only three.

CompEE has five fundamental tenets for its evolutionary model of individual learning:

- 1) Evolution is also a knowledge gaining process.
- 2) A living thing is a hierarchically organized system, each level of which is defined by the presence of an evolutionary process.
- 3) A living thing increases its fitness, through adaptation, by internally modeling stable features of the environment at several different levels.
- 4) Knowledge (theories) grows and develops in the same way as the embedded knowledge (genes) found in animals and humans.
- 5) There is an increasing fit, through adaptation, between a living thing's hypotheses about the world and the way the world actually is.

In accordance with these tenets, CompEE attempts to show:

- 1) Learning occurs at many different levels.
- 2) Evolution is a universal process for knowledge gain, and that the growth of symbolic knowledge can be achieved using evolution.
- 4) That stable, complex structures, including knowledge, can be evolved as hierarchies.
- 5) That the 'preprogramming' problem of Machine Learning- the hand-coding of knowledge and control parameters - can be managed.

CompEE describes the structure and growth of knowledge - at many different levels - in terms of biology and evolution. CompEE's multi-level approach takes a population of SimOrgs, places them in an environment, and has them learn - at the appropriate level - useful behaviors and features of the environment.

In CompEE, a SimOrg is represented by three levels, each having an influence on the others that is the strongest for adjacent levels. The three levels are as follows:

- 1) The Genotypic Level
- 2) The Phenotypic Level
- 3) The Cognitive Level

Knowledge that must be obtained immediately - possibly in order to keep a SimOrg alive - must be learned at the cognitive level. This is an example of individual learning. New capabilities that enable a SimOrg to perform better in its environment must be determined at the phenotypic level. This is an example of phenotypic learning. The parameters of these capabilities must be determined at the genotypic level. This is an

example of species learning. Each level acquires information that is important for the next one - without a brain, there would be no theories, and without genes there would be no brain.

Each of CompEE's levels defines an associated population of entities for evolution to operate on. The genotypic level has a population of genomes. The phenotypic level has a population of SimOrgs. A SimOrg's cognitive level considers its knowledge as its 'population'. The population at each level is subjected to its own mechanisms of variation, selection, and reproduction. However, what evolves is dependent upon the other levels' populations.

In CompEE, each environment has only one genotypic level population and one phenotypic level population, but many populations at the cognitive level. Each SimOrg in the environment maintains its own, separately evolving, knowledge population. A SimOrg's knowledge evolves independently of other SimOrgs' knowledge, as they don't communicate with one another. The reason for this restriction is that we are interested in the accumulation of knowledge of the individual, not of the species.

Using a new model of reasoning called Evolution-Based Reasoning, CompEE shows that the growth of symbolic knowledge can be achieved with evolution. Similar to Case-Based Reasoning, EBR attempts to reuse past knowledge. However, in contrast to CBR, EBR takes learning and adaptation to be integral components of the reasoning method. To solve problems, EBR assembles existing schemata and compares them with memories as to their appropriateness. If no 'good' schema is found to solve a problem, existing schemata are adapted to fit the problem. Schemata are never blindly assembled; they are always guided by background knowledge that limits their range of variability. The best 'fitting' knowledge, i.e. the one with the best 'match' to the problem, is selected as the solution. The solution, however, may not appear right away. It may take several generations and variations of competing schemata to determine one for the current situation.

It is ultimately how successfully the chosen schema works that determines how and if it will be used again in the future.

CompEE evolves complex knowledge structures using hierarchies. GAs derive most of their power from the implicit processing of schemata. Schemata can be viewed as stable subsystems that are continually combined, forming larger stable subsystems. It is important to note, however, that schemata, as described in the literature, function only as theoretical constructs: they are not processed explicitly. We have previously described [Deugo and Oppacher, 1992] the benefits of exploiting the power of schemata directly, not just by using them to analyze the performance of a GA, but by using them as explicit components of the representation. We found that, for the Traveling Salesman Problem domain, the direct use of schemata provided a much more natural representation and made the application of genetic operators easier. Empirical results showed that the representation and operators quickly provide good results. These benefits are also seen when CompEE uses simple tree-like structures to represent the genome and high-level knowledge structures such as plans and analogies.

In CompEE, hierarchies are used to represent a genome, a SimOrg, and knowledge. Strong hierarchies are treated as stable, nondisruptable schemata. These hierarchies are combined, in turn, into higher-level hierarchies. We have found that, as did Simon [Simon, 1973]:

'The time required for a complex system to evolve by a process of natural selection is very much shorter if the system is itself comprised of one or more layers of stable component subsystems than if its elementary parts are its only stable components.' [Simon, 1973]

Determining the parameter values of a genetic algorithm is as difficult as determining its fitness function. Similarly, how can one know, a priori, what adaptation rules a CBR system will require, or what knowledge an Explanation-Based Learning system will need? CompEE manages this 'preprogramming' problem of Machine Learning - the hand-coding of knowledge and control parameters by having the system determine its own parameters, rules, and knowledge.

CompEE attacks the 'brittleness' problem of symbolic Artificial Intelligence systems: the inability to operate outside narrowly defined domains. By not preprogramming knowledge or control parameters, and by making evolution and adaptation essential components of the model, CompEE is a much more flexible model for learning.

CompEE's genotypic and phenotypic levels correspond to AL's and Animat's genotype and phenotype level. These levels fix parameters and 'knowledge' that cannot be learned at the cognitive level. However, the cognitive level makes better use of the traits provided by the other two levels.

CompEE is related to CBR, especially by the way EBR manipulates previously used schemata in constructing and proposing new solutions. The main difference between EBR and CBR is, rather than just 'tweaking' past solutions with hand-coded rules or methods as CBR does, EBR defines knowledge structures - schemata - and operators for the explicit purpose of adapting knowledge. CompEE, using evolution, takes adaptation seriously.

CompEE is related to GAs but manipulates hierarchical structures rather than fixed size sequential bit strings.

CompEE is also related to EE. The work in EE has gone virtually unnoticed by the GA, AL and Animat communities. This is an omission that we wish to correct. EE offers many ideas on the topics of the evolution of knowledge, learning, and behavior, and should not be overlooked. Although EE has failed to advance detailed, empirically testable ideas, CompEE, with its Smalltalk implementation, does just that.

3.3 CompEE's SimOrg

A SimOrg is implemented as an evolutionary machine. Externally, it is composed of a collection of sensors and is capable of performing actions suitable for its environment. Internally it is composed of a hierarchy of three dependent levels: the genotypic level, the phenotypic level, and the cognitive level. The structures at each level are subjected to greatly simplified simulations of the Darwinian Theory of Evolution by Natural Selection: 1) fitness proportional reproduction, and 2) genetic operators such as crossover and mutation.

It is easy to see that in most biological systems, the underlying structure is hierarchical. For example, molecule -> cell -> tissue -> organ -> organism, is one such hierarchy. This is an example of a containment or structural hierarchy. Evolution is often thought of as acting on structures that are organized with a structural hierarchy [Eldredge and Salthe, 1984; Hull, 1981,1988]. The main feature of this type of hierarchy is what Simon [Simon, 1973] refers to as the Chinese Box characteristic. The notion is that if we view any structure in the hierarchy we find that it is composed of structures located at the next lower level in the hierarchy, and those structures, if opened, would be seen to be composed of structures located at the next lower level, and so on.

In a structural hierarchy, stable lower-level structures with understood characteristics, e.g. form, function and interface requirements, are combined to form higher-level structures. This information constrains how higher-level structures can use the sub-structures to build complex ones. For example, molecules are combined to form cells and tested, and the successful cells combined into tissues and tested, etc. The construction process builds higher-level structures by assembling stable lower-level ones. This is not a blind process; it is constrained and guided by the properties of lower-level structures.

Many organizations such as governments and businesses are also hierarchically organized. At the top of the hierarchy are the bosses or the central government, in the middle are the managers or the district governments, and at the bottom are the workers. These are examples of control hierarchies [Pattee, 1973]. This type of hierarchy orders the entities within the hierarchy on the basis of a controller/controllee relationship.

The above discussion imposes two requirements on our hierarchical representation of a SimOrg and the structures found in its different levels:

1. Structures found at higher levels of the hierarchy are constrained by, and conform to, the lower level structures and their physical laws.
2. The end achievements of the higher level structures require, for their implementation, the lower level structures.

The fact that higher level structures use or are constructed from lower-level structures impose two additional requirements, noted by Simon [Simon, 1969]:

3. At each level in the hierarchy, its structures must be stable (exhibit temporal consistency) .
4. At each level in the hierarchy, its structures must be adaptable.

The stability of lower level structures provides the consistency over time for higher level structures to evolve. If lower level structures are unstable, it is difficult for stable, higher level ones to form - their building blocks keep falling apart.

Structures that are adaptable can perform the same function in many contexts. This feature enables lower level structures to be used as parts of many different higher level

structures. Being a part of many higher level structures increases a lower level structure's utility.

CompEE views evolution operating on SimOrgs represented as a multi-level control hierarchies, shown in figure 3.1.

Figure 3.1 SimOrg Hierarchy

Associated with each level of the hierarchy is a population of structures: genomes, somatic structures, and knowledge. Each structure is represented as a structural hierarchy. As noted by Hunkapiller [Hunkapiller *et al.*, 1982] for genes, but extended here to include all of a SimOrg's structures not just those at the genotypic level, three important features result from the use of a hierarchical representation :

- 1 *The units of evolutionary information can be seen as building blocks and hierarchical assemblies of building blocks.* Complex structures, such as the genome and knowledge, are seen to be comprised of one or more layers of similar, less complex structures.
2. *A hierarchical representation provides the potential for dynamic flexibility in organization.* The ability for multi-building block structures to be treated as coordinated wholes increases the flexibility and abstraction of the organization. The end result is that building blocks allow for the formation of complex, dynamic structures.
3. *A hierarchical organization leads to rapid and extensive variation in the resulting structures.* Stable lower level structures are essential for the development of higher level components, and, as implied, can be parts of many higher level components. Therefore, higher level components can form quickly using a predefined collection of well designed parts. The speed and size of variations is also increased by manipulating entire subsections of the hierarchy rather than just single nodes, although this too is permitted.

In summary, a single control hierarchy is used to coordinate CompEE's three levels: the genotypic level, the phenotypic level, and the cognitive level. At each level, a different population of structures exists. There is a population of genomes, each composed of a hierarchy of genes, at the genotypic level; there is a population of SimOrgs, composed of hierarchically arranged somatic structures, at the phenotypic level; and, at the cognitive level there is hierarchically arranged knowledge, composed of plans or hypotheses, analogies, and concepts. Although each level in a SimOrg's control hierarchy is independent (with respect to the evolution), each level is dependent on the others for its operation.

3.4 Chapter Summary

Computational Evolutionary Epistemology, a biologically motivated approach to cognitive modeling and machine learning, proposes three different classes of learning, based on changes in the internal organization of a phenotype or animat: species learning, development learning, and individual learning. CompEE is a form of EE for the individual, and treats cognition as a biological phenomenon by extending biological evolution to those traits of animals and humans that are the biological substrates of cognition. Its multi-level approach takes a population of SimOrgs, places them in an environment, and has them learn - at the appropriate level - useful behaviors and features of the environment. At each level, a population of structures: genomes, somatic structures, and knowledge, is subjected to greatly simplified simulations of the Darwinian Theory of Evolution by Natural Selection: 1) fitness proportional reproduction, and 2) genetic operators such as crossover and mutation.

Chapter 4

The Genotypic Level

In CompEE, evolution operates on a SimOrg represented as a multi-level control hierarchy. The most primitive of these levels is called the genotypic level. It is at this level that a SimOrg's description is stored and adapted, and without it the other levels would fail to operate. A single genome contains this description, which defines a SimOrg's structures and control parameters, e.g., its memory capacity, its life span, the mutation rate, the sensitivity of its sensors, and its motivational drive priorities. It is the genotypic level's responsibility to learn the correct values and organizations of such parameters and priorities that enable a SimOrg and its future offspring to survive in the environment. These two responsibilities represent a step towards a parameterless system. We are not proposing that the system create its own parameters, but it will identify the values for the ones we give it, and determine the best organization for them.

Since learning at this level is not immediate, e.g., it may take several generations to locate combinations of settings and arrangements of the genome's parameters that prove useful in the environment, it is the species that benefits from the changes in the genome. Therefore, the genotypic level is concerned with species learning.

In this chapter we discuss the genotypic level in detail, describing the structure of the genome, how it is adapted using the process of evolution, and how its evolution results in species learning. Of particular significance is our combination of hierarchies with non-

binary alphabets to represent the genome instead of the often used fixed size, sequential bit strings, and how the organization and contents of the genome's parameters and drive priorities are evolved. The problem of determining the best coding⁶ for a genome a priori is an impossible task in any environment where the details of the selection process can't be identified - a serious problem to consider when using a GA. Our representation for the genome and the methods for its adaptation are created to provide a solution to this problem, and are directly applicable in a GA. Before describing the genome and its components, the following section describes the motivation for our choice of hierarchies to represent the genome's components.

4.1 The Evolution of Hierarchies

In the areas of GAs, AL and Animats, the genetic material is often represented as a chromosome [Goldberg, 1989] containing a fixed number of genes or features which can assume certain values called alleles. The collection of values provides the genetic information required for constructing a phenotype (SimOrg) or for determining a possible solution to a given problem. When represented as a fixed size, sequential sequence of genes [Rizki & Conrad, 1986], the chromosome is a static structure: every gene's locus is fixed, and each gene is used to embody a physical characteristic or behavior of the phenotype in which it resides. To simplify the encoding further, GAs operate with alleles of 0 and 1.

The simplification of a fixed size, sequential sequence of genes is a result of one of the fundamental principles of GA: the principle of meaningful building blocks:

⁶ Arrangement of the genes.

'The user should select codings so that short, low-order schemata are relevant to the underlying problem and relatively unrelated to schemata over other fixed positions.' [Goldberg, 1989]

Schemata provide a compact way to express similarities among chromosomes, and are themselves chromosomes over the alphabet $\{*, 0, 1\}$. The addition of '*' - interpreted as a metalinguistic *don't care* symbol - enables schemata to denote subsets of chromosomes over the alphabet $\{0, 1\}$ which are similar at certain positions. For example, the schema *101* denotes the chromosomes: 01010, 01011, 11011, 11010, whose elements are identical in their second through fourth positions. A schema has several important properties, such as its fitness (as given by the average fitness of the chromosomes it matches), its defining length (the distance between its first and last specific chromosome positions, 2 in the example above), and its order (the number of its fixed positions, 3 in the example above).

The Schema Theorem [Holland, 1975] states that highly fit, short defining length, and low order schemata are allocated exponentially increasing trials in successive generations, under fitness-proportional reproduction, crossover, and mutation. Holland also showed that GAs possess a property of implicit parallelism: while ostensibly processing only the binary chromosomes in the population, they also process in parallel a much greater number of schemata represented in the population. Indeed, for a population of size n at least n^3 schemata are usefully processed without computational overhead.

The point of the above principle is that relevant and dependent genes should be positioned close to one another in order that operators such as crossover will be less likely to disrupt the local building blocks. This statement has two implications: that one knows the building blocks a priori, and, therefore, what genes are dependent; and that one knows a priori how many genes encode the problem. However, it is all but an impossible task to

know all gene dependencies a priori without knowledge of the selection process. For example, if one hand-codes a function to compute the fitness of a chromosome string, one can easily identify the dependent gene because the details of the computation are available. However, if one takes a reading from a mechanical device to establish fitness, take for example Goldberg's black box optimization problem [Goldberg, 1889], the details of the fitness computation are unknown. In this case, making a coding that does not violate the principle is impossible. One can only justify a coding by showing that 'it seems to work'.

Although there are many situations where GAs work even though their gene dependency information cannot be determined a priori, there are situations where they will not work without knowing this information. For example, a GA will not work if it uses a fitness function that, for optimal fitness, requires two specific genes to be located beside one another and does not provide the operators for this to occur in the chromosome. The problem of determining gene dependencies is itself a search problem, and seems well suited for the process of evolution to construct and sort out the good arrangements - the building blocks. Our approach to this problem is to evolve hierarchies that change their contents and organizations using a new tree crossover operator, which are the topics for the next sections.

4.1.1 Hierarchical Chromosomes

Can one evolve the organization of a chromosome and its contents simultaneously using a hierarchy? Exactly what is meant by a part of a chromosome? How does one measure a part's fitness? How is a chromosome and its parts processed: separately or simultaneously? Should the structure of the chromosome be a hierarchy? Is there any benefit in evolving the organization of the chromosome? Although each of these questions is important, the last one must be answered first before we proceed to answer the

remainder; for if no benefit can be found, there is no use in using a hierarchy as our representation for the contents of the genome.

As stated in the last section, in all but the simplest domains where the gene dependencies can be easily located, it is an impossible task to determine a problem's gene dependencies a priori. This is also true of the genotypic level's genome, where no information is available to help order the genome's genes. Therefore, if the fitness of the genome depends on the positioning of its genes, to achieve the best results one must allow the chromosome's genes to move and form the appropriate dependencies or else be satisfied with suboptimal results. The benefits to the genome are obvious. Allowing the genome to determine its own gene ordering, not relying on a static, hand-coded ordering, makes it a much more flexible and adaptable structure, which enables it to become, with the aid of evolution, a much fitter, stronger structure.

To help answer the remaining questions we construct and discuss four different GAs. Each one uses a different chromosome organization, but all share a common fitness function. The first GA uses a chromosome constructed as a linked list which is ordered according to the correct gene dependencies. The second GA uses a chromosome constructed as linked list in which the genes are randomly arranged. In these two organizations the gene at the head of the list is called the root gene. The third GA uses a chromosome constructed as a rooted tree, with genes randomly arranged and represented by the nodes of the tree and their dependencies represented by its links. The tree is considered complete: every one of the chromosome's genes is present in the tree. The fourth GA uses a chromosome similar to the third organization except that not all genes may be present in the tree. In each organization a gene's value can be either 0 or 1.

The fitness function used by all organizations is dependent on the ordering of a chromosome's genes and their values, and is described as follows:

Fitness

$\wedge$ fitness := (self rootGene GeneFitness) / (2 * numberOfGenesInChromosome).

GeneFitness

| sum |

sum := (self geneNumber) - (self parentGene geneNumber).

"Only reward a gene if its geneNumber is greater than its parent. The maximum fitness contribution of a gene is two, one for being in the correct order - with respect to the parent gene - and one for having a gene value of one"

(sum >= 0)

if True: [sum := 1 + self GeneAllele]

if False: [sum := self GeneAllele].

self linkedGenes do: [:childGene]

sum := sum + childGene GeneFitness].

$\wedge$ sum

In the above pseudo code, self refers to a chromosome. The fitness of a chromosome is calculated as the fitness of its genes divided by two times the number of genes in the chromosome. The reason for the division is to establish a fitness value between zero and one, since the maximum fitness value of the genes is two times the number of genes in the chromosome.

The recursive function GeneFitness works through a chromosome's linked gene structures (lists or trees) starting at the root gene to establish the total fitness of the chromosome. If a gene's preceding gene's (in the case of a linked list) or the parent gene's

(in the case of a tree) gene number is less than its own⁷, a value of 1 is added to the fitness total. Next, the gene's value is added to the total. Therefore, the contribution of each gene to the overall fitness can be either 0, 1, or 2, as shown in table 4.1. The fittest chromosome will have all its gene values set to 1 and its genes ordered from the smallest to the largest gene number. The fitness function imposes a dependency on the chromosome; to receive the maximum fitness computation, its genes must be in the correct order. Since this dependency is unknown to the chromosome, the chromosome must discover it and the correct gene settings to achieve maximum fitness.

Correct Order	Gene Value	Fitness Contribution
yes	0	1
yes	1	2
no	0	0
no	1	1

Table 4.1 Gene Fitness Contribution

As an example of calculating a chromosome's fitness, consider the following linked list, four gene example. The first gene is located at position 1, the second at position 2 and so on. The gene values are as follows: 0101. In this case, the function GeneFitness would return 6: 1 + 2 + 1 + 2. Therefore, the fitness of the chromosome is 0.75 (6/8).

Gene Number:	1 <-	2 <-	3 <-	4	
Gene Allele:	0	1	0	1	
Fitness Contribution:	1	2	1	2	GeneFitness Total: 6

⁷ The root gene's parent's gene number is the same as the root.

As another example of calculating a chromosome's fitness, consider the following random linked list, four gene example. In this example, the positions of the genes are randomly assigned in the linked list. In this case, the function GeneFitness would return 5: $1 + 2 + 0 + 2$. Therefore, the fitness of the chromosome is 0.625 (5/8).

Gene Number:	3 <-	4 <-	1 <-	2	
Gene Allele:	0	1	0	1	
Fitness Contribution:	1	2	0	2	GeneFitness Total: 5

In this example gene number 1 is penalized since its parent gene's number, which is 4, is greater than its own.

As a final example consider the following tree, four gene example. Gene 2 is the root gene with value 1 and two children: gene 1 with value 0 and gene 3 with value 0. Gene 1 has one child: gene 4 with value 1. In this case the function GeneFitness would return 5: $0 + 2 + 1 + 2$. Therefore, the fitness of the chromosome is 0.625 (5/8).

Figure 4.1 Tree Chromosome

Gene Number:	4 ->	1 ->	2 <-	3	
Gene Allele:	1	0	1	0	
Fitness Contribution:	2	0	2	1	GeneFitness Total: 5

In this example gene number 1 is penalized since its parent gene's number, which is 2, is greater than its own.

The initial fitness of a subtree is computed like that of a tree. However, because a subtree does not have all of its genes present, its fitness value must also be made to reflect this fact. This change to the fitness computation is discussed in section 4.1.1.4.

In the following subsections, we describe the results of running a GA with each of the above organizations using the above fitness function, and any new operators required to support them. We then summarize by describing the approach that best suits our goal for evolving both the organization and contents of the genome.

4.1.1.1 Sequential Genetic Algorithm

The sequential GA's (SGA's) chromosome is organized as a linked list of genes. Gene number one is the first element in the list, gene number two is the second element in the list, and is linked to the first one, and so on. As the genes are already in the correct order, e.g., each gene's preceding gene (or parent gene) has a smaller number than itself, all the SGA must do is determine the correct values for the genes - 0 or 1. This also implies that the fitness value returned for any chromosome is at least 0.5 because each gene in the correct order contributes one to the chromosome's fitness.

The SGA uses the two-point crossover operator [Syswerda, 1989] for reordering a chromosome. The two-point crossover operator chooses two points (links) in a chromosome at random, and exchanges the genes between them with those of another

chromosome between the same two points. For example, given the following two chromosomes, A and B, and crossover points 2 and 5, the application of the two-point crossover to A and B produces the new chromosomes A' and B' as follows:

A:	111111111	A':	1000011111
B:	000000000	B':	0111100000

One important feature of this operator, especially for the SGA, is that its application does not cause any change in the gene ordering of the resulting chromosomes. As shown by the example, gene number 2 in chromosome A is still gene number 2 in chromosome B'. The operator's application results in only an exchange of gene values while maintaining gene ordering.

The fact that the SGA must only determine the correct gene values reduces the fitness function to that of the linear function defined as follows: $f(x) = c / n$. The value of the function is simply the number of 1 bits, c , in the input chromosome vector x divided by the number of bits, n , in the vector. This function presents no difficulty because one hillclimb from anywhere in the space will find the maximum value [Ackley, 1987].

The results shown in figure 4.7 prove this to be true. Figure 4.7 shows the median test run (out of 11) for the SGA to evolve the correct values of chromosomes of sizes of 5, 10, 20, and 40. The population sizes used by the SGA are a linear function of the size of the chromosome, defined as follows: $\text{PopulationSize}(x) = 2 * \text{size of } x$. The population size is two times the size of the chromosome vector x . As the graph indicates, the SGA has no problem in finding the best gene setting and quickly converges to a solution for varying chromosome sizes. This result is to be expected and establishes a baseline for the remaining GA test runs using different chromosome organizations. However, remember

that the SGA did not have to establish the gene ordering; that was done a priori. It is our goal to produce a GA that evolves both the best organization and values of its chromosome.

4.1.1.2 Random Sequential Genetic Algorithm

The random GA's (RGA's) chromosome is organized, liked the SGA, as a linked list of genes. However, genes are randomly assigned a location in the liked list. Therefore, gene number one could be the last element in the list, gene number two could be the middle element in the list and possibly preceded by the last one, and so on. As the genes are not in the correct order, e.g., each gene's preceding gene (or parent gene) may not have a smaller number than itself, the RGA must determine the correct gene ordering and values. This also implies that the fitness value returned for any chromosome is between 0 and 1, because each gene may contribute a value between zero and one to the chromosome's fitness.

Like the SGA, the RGA uses the two-point crossover operator. However, this causes a problem for it. For example, given the following two chromosomes, A and B (with genes encoded as tuples: [value:gene-number], e.g., 0:3 means gene 3 has value 0), and crossover points 2 and 5, the application of the two-point crossover produces the two chromosomes A' and B' as follows:

A: 1:1, 1:3, 1:5, 1:7, 1:9, 1:2, 1:4, 1:6, 1:8, 1:10

B: 0:7, 0:6, 0:5, 0:1, 0:9, 0:3, 0:10, 0: 4, 0:8, 0:2

A': 1:1, 0:6, 0:5, 0:1, 0:9, 1:2, 1:4, 1:6, 1:8, 1:10

B': 0:7, 1:3, 1:5, 1:7, 1:9, 0:3, 0:10, 0: 4, 0:8, 0:2

A good feature of this operator for the RGA is that its application causes a change in the gene ordering of the resulting chromosomes. Since the chromosome's gene ordering is

randomly initialized, and only 1 in $n!$ gene orderings is correct, the RGA requires an operator that will alter the gene ordering. However, the operator's problematic feature is that its application results in both a duplication and loss of genes in the resulting chromosomes. For example, chromosome A' has genes 1 and 6 present twice, and for both values 0 and 1. Also, genes 3 and 7 are missing from it. Without some other operator for the RGA to correct these problems, the RGA seems doomed to fail. This was in fact the end result. None of our test runs ever converged to the correct answer, and, therefore, these results have been omitted from figure 4.7. The main problem is that after a few generations the population is filled with chromosomes with missing genes. Since mutation only changes the gene values and not the gene numbers, the missing gene can not return to the population. After this point there was no way that the RGA could converge.

The RGA does not work because it does not provide an operator, such as inversion, to move dependent genes close to one another. One of the problems with using a unary operator such as inversion is that the new chromosome it produces does not gain from the experiences of other chromosomes. Since it was next to impossible for the RGA to produce the correct chromosome using two-point crossover and mutation - there was always a remote chance that the RGA could do it, however, it never happened in our experiments - the only other way for the RGA to produce it was to stumble upon it in the very first generation. The probability of this happening is $1 / 2^n * n!$, and was unlikely⁸. Therefore, we conclude that the RGA is not a successful approach for evolving the organization and values of a gene dependent chromosome.

⁸ $n!$ different gene orderings, each with 2^n different settings.

4.1.1.3 Tree Genetic Algorithm

The tree GA's (TGA's) chromosome is organized as a rooted tree of genes. Like the RGA, genes are randomly placed in the tree, organized initially as a binary tree. Therefore, gene number one could be a leaf node, gene number two could be an internal node, the last gene could be the root node, and so on. As the genes are not in the correct order, e.g., each gene's parent gene may not have a smaller number than itself, the TGA must determine the correct gene ordering and values. This also implies that the fitness value returned for any chromosome is between 0 and 1 because each gene may contribute a value between zero and one to the chromosome's fitness.

Although initially constructed as a binary tree, the tree will become a n -ary tree as a result of its crossover operator. The TGA uses a new crossover operator called: The Tree Crossover Operator (TreCoP), defined as follows:

TreeCrossoverWith: aChromosome

```
! myTreeGeneRoot mySubTree matesSubTree parentSubTree !
```

```
"Access the root of the gene Tree"
```

```
myTreeGeneRoot := self genes.
```

```
"Select a node/gene in my geneTree using a weighted selection function that  
selects the root of the tree with probability 0.5, and all other nodes equally  
probable"
```

```
mySubTree := myTreeGeneRoot weightedSelectARootGeneAtRandom.
```

```
"Select a node/gene in the mating chromosome's geneTree using a weighted  
selection function that selects a child node of the root of the tree with
```

probability 0.5, and all other nodes equally probable"

matesSubTree := aChromosome genes weightedSelectAChildGeneAtRandom.

" Join the two selected subtrees, one from the current chromosome and
one from the mating chromosome"

(mySubTree isRoot)

ifTrue: [

" If my subtree was in fact the entire tree, then use my mate's subtree as the
new rooted gene tree"

myTreeGeneRoot := matesSubTree]

ifFalse: [

" Replace my selected subtree with that of my mates subtree"

parentSubTree := mySubTree returnParentTree.

parentSubTree removeChild: mySubTree;

addChild: matesSubTree.

"myTreeGeneRoot now has the new subtree attached"].

" Clean duplicate and missing node/gene problems that result
from joining subtrees "

myTreeGeneRoot removeDuplicateGenes.

myTreeGeneRoot addMissingGenes.

"Update my genes to have the new geneTree"

self genes: myTreeGeneRoot .

The crossover operator takes two chromosomes, represented as trees, and updates the original one, referred to as 'self' in the pseudo-code, with the product of the crossover of the two rooted gene trees. First a node (gene) is selected in the original tree. The selection process is weighted to select its root node with probability 0.5, and all other nodes equally weighted. Next, a node in the mate's tree is selected. The selection process is weighted to select a child node of its root with probability 0.5, and all other nodes equally weighted. Next, the original's subtree is replaced by the mate's subtree, unless the original subtree selected is the entire tree, and in this case the mate's subtree becomes the new rooted tree. The crossover operation may cause genes to be duplicated or deleted from the resulting tree. Therefore, the next operation performed is the removal of duplicate genes and the replacement of missing ones. Finally, the constructed tree replaces the original chromosome's gene tree.

The reason for using the weighted selection functions is simple. They result in a constant pulling of genes towards the root of the tree which is desirable for reorganizing the genes in the tree. Since the original subtree is often the root (at least 50% of the time), the mate's subtree will replace it. Since the mate's subtree is often a child of the root (at least 50% of the time), and since gene trees tend to be similar as the population converges to a solution (meaning that the two gene trees are quite similar, if not identical), this results in elevating a child to the status of the root of the tree - a pulling up of the nodes in the tree from a child of the root.

What is also desirable about this operator is that the new tree it produces resembles the two parent trees. To do this, some care must be taken removing duplicate genes and restoring missing ones. The function `removeDuplicateGenes` performs its gene removal task in a depth first manner. Once a duplicate gene is detected, it is removed from the tree and its children added as the children of the removed node's parent. For example, removing gene 2 from the left tree of figure 4.2 results in the right tree of the figure.

If the gene being removed is at the root of the tree, one of its children is selected at random to become the new root of the tree. The remaining children of the old root are then added as the children of the new root. For example, removing gene 1 from the left tree of figure 4.3 results in the right tree of the figure.

Figure 4.2 Duplicate Node Removal

Figure 4.3 Duplicate Root Node Removal

The function `addMissingGenes` adds missing genes into the new tree in the same location as they were located in the original tree⁹. Take for example the tree shown in figure 4.4, if gene 3 was a child of gene 2 in the original tree, and it is missing from the resulting tree, it is added as a child of gene 2 in the resulting tree. If this is not possible, because the parent gene - gene 2 - is not yet present in the resulting tree, the gene is randomly added to one of the leaf nodes. By ordering the missing genes in the way they are visited by a breadth first search of the original tree, this problem is avoided. The reason being if two dependent genes are missing from the resulting tree, the parent gene is always added first, ensuring that the child can always be attached to it and henceforth in its proper location.

Figure 4.4 Missing Node Addition

⁹ Only those genes missing from the original tree, not the mate's tree, are added.

The important feature of the TreCoP and its two supporting functions: `addMissingGenes`, and `removeDuplicateGenes`, is they attempt to maintain the same spirit of organization in the resulting chromosome as was found in the parent chromosomes, while also managing to change and reorganize the genes into a combination of the two. Since a TGA's chromosome gene ordering is randomly initialized it is important that the TreCoP search for good gene orderings. However, it must not mix the existing chromosomes so much as to lose any information that has been gained from past explorations. The operator must simultaneously exploit old gene orderings while exploring new ones - the hallmark of GAs.

Does the TGA work? Does it locate the best gene ordering and values? The answer is yes. Figure 4.7 again shows the median test run (out of 11) for the TGA to evolve the correct gene ordering and values for chromosomes of size 5, 10, 20, and 40. Although not as fast the SGA, the TGA has no problem converging to a solution for varying chromosome sizes. The extra time required for the TGA to converge to a result, compared to the SGA, was to be expected. It must evolve both the organization and values of the genes. The SGA only evolves the values.

To get a rough estimate on the number of different trees in the TGA search space consider the case of oriented trees. Oriented trees are not different when they differ only in the respective ordering of subtree nodes, which is the case in the TGA's chromosome. For oriented trees with n labeled vertices (genes), there are n^{n-1} distinct oriented trees [Knuth, 1973]. In addition to this space, the TGA searches the space of 2^n different gene values. Therefore, the TGA searches a space of size approximately $2^n * n^{n-1}$.

Even with this enormous search space the TGA works well, and is one viable approach to evolving a chromosome's organization and gene values simultaneously. Can we do better?

4.1.1.4 Subtree Genetic Algorithm

The subtree GA's (STGA's) chromosome is organized, similar to the TGA, as a rooted tree of genes. Unlike the TGA, the STGA's chromosome does not require the presence of all genes - at which time it is considered as a chromosomal subtree. In fact, a STGA's chromosome is usually missing some of its genes. However, a STGA's chromosome does maintain that no gene is duplicated in a subtree.

Initially, a random subset of the genes is selected for inclusion in the chromosome, constructed as binary tree. The chromosome is then subjected to a new crossover operator called: The SubTree Crossover Operator (STreCoP). As a result of the application of this operator - merging the genetic information of two source chromosomes into one - the chromosome's subtree changes from its initial form to that of an n -ary tree. In addition, the operator also adds and deletes genes from the subtree, forming new subtrees or complete¹⁰ trees.

The pseudo-code for the STreCoP is identical to that of the TreCoP. However, its behavior is slightly different in one respect. As in the TreCoP, the function `addMissingGenes` adds missing genes to the newly generated tree in the same location they were located in the original tree. However, this procedure adds only those genes that were missing from the original subtree, not the mate's subtree. This feature makes little difference in the TreCoP; the original tree has all of its genes, and so will its offspring. However, since a STGA's chromosome may not have all of its genes, unless those missing genes are crossed over from the mate, the missing genes will not be added by the procedure.

Another difference between the TGA and the STGA is in the way the STGA measures the fitness of a chromosome. Since STGA will have both complete and incomplete

¹⁰ All genes present.

chromosomes, each of varying size, a method is required to establish the fitness of each organization relative to the others.

The fitness of a chromosome is initially calculated as described in section 4.1.1: the fitness of its genes divided by two times the number of genes in the chromosome. This method guarantees that all chromosomes have a fitness value between 0 and 1. Next, the fittest chromosome is selected from the STGA's population and its size - the number of genes - is used to rank the remaining chromosome relative to it. It is not the case that the best chromosome is always the largest. For example, a two node chromosome with both genes set to 1 and arranged in the correct order will have a fitness of 1. A ten node chromosome with all genes set to zero and in the wrong order will have a fitness of 0. Using the fittest chromosome's size ('relativeSize' below), we calculate the fitness of all other chromosome as follows:

RelativeFitness: relativeSize

(self size > relativeSize)

ifTrue: [

"If my chromosome size is greater than the best chromosome's size, then

set my fitness to an amount proportional to the ratio of the sizes"

^ fitness := self fitness * relativeSize / self size]

ifFalse: [

"If my chromosome size is less than the best chromosome's size, then

set my fitness to an amount proportional to the ratio of the sizes"

^ fitness := self fitness * self size / relativeSize]

The fitness of a chromosome is changed proportional to the ratio of its size and the size of the fittest chromosome. Chromosomes that have more or less genes than the best

chromosome will see their fitness values decrease. Chromosomes that have the same number of genes will see no change in their fitness values. The effect of this fitness function is to keep the STGA working at one level - size of chromosome - until good subtrees - building blocks - are produced. These subtrees are then the material for the construction of the next level. Once the next level is established, i.e., the chromosomes at that level have superior fitness, the STGA's effort is then concentrated there, providing the subtrees for the next level, and so on.

For example, consider the following median run of a STGA using chromosomes with 10 genes. From left to right, the columns indicate the generation number, the fitness of the best individual in the population, the median fitness value of the population of twenty chromosomes, the fitness of the worst individual, and the fitness, chromosome organization and gene values of the second best individual in the population (7/5:1 indicates that gene 7 is a child of gene 5 and has a value of 1).

0	1.0	0.36	0.01	[1.0]	5/5:1, 7/5:1
1	1.0	0.40	0.01	[0.66]	4/4:1, 5/4:1
2	1.0	0.65	0.12	[1.0]	5/5:1, 6/5:1, 7/5:1
3	1.0	0.59	0.15	[0.75]	3/3:1, 5/3:1, 10/5:1
4	1.0	0.63	0.28	[1.0]	1/1:1, 3/1:1, 9/1:1, 10/1:1
5	1.0	0.61	0.33	[1.0]	1/1:1, 3/1:1, 9/1:1, 10/1:1
6	1.0	0.59	0.26	[0.84]	1/1:1, 9/1:1, 2/1:1, 8/1:1, 3/8:1, 10/8:1
7	1.0	0.61	0.25	[0.84]	1/1:1, 3/1:1, 5/3:1, 7/5:1, 6/1:1, 4/6:1
8	1.0	0.59	0.22	[0.85]	1/1:1, 3/1:1, 5/3:1, 7/5:1, 2/1:1, 4/2:1
9	1.0	0.69	0.48	[0.86]	1/1:1, 3/1:1, 8/1:1, 4/1:1, 7/4:1, 5/7:1, 9/5:1
10	1.0	0.66	0.34	[1.0]	1/1:1, 3/1:1, 5/3:1, 7/5:1, 2/1:1, 10/2:1, 4/2:1
11	1.0	0.73	0.40	[1.0]	1/1:1, 3/1:1, 5/3:1, 7/5:1, 9/5:1, 2/1:1, 4/2:1
12	1.0	0.71	0.44	[1.0]	1/1:1, 3/1:1, 5/3:1, 7/5:1, 9/5:1, 2/1:1, 4/2:1
13	1.0	0.61	0.32	[1.0]	1/1:1, 3/1:1, 5/3:1, 7/5:1, 9/5:1, 2/1:1, 4/2:1
14	1.0	0.58	0.25	[0.76]	3/3:1, 5/3:1, 7/5:1, 10/5:1, 9/5:1, 8/3:1, 2/8:1, 4/2:1
15	1.0	0.60	0.34	[0.86]	1/1:1, 2/1:1, 4/2:1, 10/1:0, 3/1:1, 5/3:1, 7/5:1
16	1.0	0.62	0.34	[1.0]	1/1:1, 3/1:1, 5/3:1, 7/5:1, 9/5:1, 2/1:1, 4/2:1
17	1.0	0.58	0.40	[0.86]	1/1:1, 3/1:1, 5/3:1, 7/5:1, 9/5:0, 2/1:1, 4/2:1
18	1.0	0.57	0.34	[1.0]	1/1:1, 3/1:1, 5/3:1, 7/5:1, 9/5:1, 2/1:1, 4/2:1
19	1.0	0.73	0.40	[1.0]	1/1:1, 3/1:1, 5/3:1, 7/5:1, 9/5:1, 2/1:1, 4/2:1
20	1.0	0.71	0.44	[0.87]	1/1:1, 3/1:1, 5/3:1, 7/5:1, 6/5:1, 2/1:1, 4/2:1
21	1.0	0.68	0.44	[0.87]	1/1:1, 3/1:1, 5/3:1, 7/5:1, 9/5:1, 2/1:1, 6/2:1
22	1.0	0.68	0.39	[1.0]	1/1:1, 3/1:1, 5/3:1, 9/5:1, 10/5:1, 2/1:1, 8/2:1, 4/2:1
23	1.0	0.75	0.53	[1.0]	1/1:1, 2/1:1, 8/2:1, 10/2:1, 3/1:1, 5/3:1, 7/5:1, 6/5:1
24	1.0	0.73	0.46	[1.0]	1/1:1, 2/1:1, 8/2:1, 10/2:1, 6/1:1, 4/1:1, 5/4:1, 7/5:1
25	1.0	0.71	0.31	[1.0]	1/1:1, 3/1:1, 5/3:1, 7/5:1, 6/5:1, 2/1:1, 8/2:1, 4/2:1
26	1.0	0.66	0.25	[0.90]	1/1:1, 3/1:1, 5/3:1, 10/5:1, 7/5:1, 2/1:1, 9/2:0, 6/2:1, 4/2:1, 8/2:1

Looking down the right hand side of the above STGA run one can see how the size of the second best chromosome changes. Once the STGA produces a good chromosome of a particular size it remains at that level until more good ones of the same size can be produced for the construction of chromosomes for the next level. Initially, this occurs quickly. The reason is that there are very few subtrees to consider at the initial levels. However, at the final levels, as the subtrees becomes larger, the switching of levels is not as fast because of the increased number of subtrees that have to be considered before proceeding to the next level. In the above test run, it took twenty-six generations for the STGA to produce the complete chromosome shown in figure 4.4: each gene in the correct position, and each with the correct value (1).

Figure 4.5 Complete Chromosome Produced by the STGA

Does the STGA work? Does it locate the best gene ordering and values? The answer is obviously yes. The surprising result is that it works faster than the TGA. Figure 4.7 again shows the median test run (out of 11) for the STGA to evolve the correct gene ordering and values for chromosomes of sizes 5, 10, 20, and 40. Although not as fast the SGA, the

STGA has no problem in finding the best gene setting and quickly converges to a solution for varying chromosome sizes. The extra time required for the STGA to converge to a result is to be expected. It must evolve both the organization and values of the genes. The SGA only evolves the values.

Previous research into the Traveling Salesman Problem (TSP) [Deugo & Oppacher, 1992] achieved similar results. In that work we developed an efficient schema-based representation and operators for solving the TSP. A schema was used as an explicit part of the representation rather than just a theoretical construct. In that domain we found that the direct use of schemata provided a much more natural representation and made the application of genetic operators easier. For example, by using schemata, incomplete tours were permitted, and, because operators did not have to form complete tours, exchange and variation of genetic material was simple. Empirical results showed that the new representation and operators quickly provide good results.

In that paper, we conjectured that explicit schema processing would reduce the complexity of problems in other domains as well. Others also believe that 'the time required for a complex system to evolve by a process of natural selection is [appears]¹¹ very much shorter if the system is itself comprised of one or more layers of stable component subsystems than if its elementary parts are its only stable components' [Simon, 1973]. Viewing a subtree as a schema (a stable component), the results of the STGA supports this conjecture.

The STGA works well, even better than the TGA, and is another, and our best, approach to evolving a chromosome's organization and gene values simultaneously.

¹¹ My own word.

4.1.2 Review of GA Organizational Strategies

Can one evolve the organization of a chromosome and its components, and their contents, simultaneously using a hierarchy? The answer is yes. Both the TGA and the STGA managed to do both. These approaches are not as fast as the SGA, but then the SGA already has the correct gene ordering. The TGA and the SGA must discover this for themselves, and this activity requires more time.

Exactly what is meant by a part of a chromosome? Since both the TGA's and the STGA's chromosome are structured as a rooted tree, a part of it is simply another tree - a subtree.

A subtree is analogous to a GA's building block: a compact way to express similarities among chromosomes. However, rather than imposing a strict ordering on all of the genes in the chromosome, as a schema does, a subtree only imposes an ordering on those genes for which the values are known. For this reason, the number of complete chromosomes represented by a subtree is more than that of a simple schema. Take for example the simple case of a schema of order $n-1$, with only one unknown gene value, e.g. 10^* . This schema represents only two strings: 101, and 100. When genes are positioned as nodes in an n -ary tree, the missing gene can be attached to any one of the existing $n-1$ nodes. Therefore, if we are using binary genes, the one subtree represents $2(n-1)$ different possible trees. In our example, the subtree would be part of four other possible trees, as shown in figure 4.6.

The difference in the number of actual strings represented by a schema and a subtree is increased as the number of missing genes increases. Therefore, if the power of a GA's search is related to types and numbers of schemata in a population, using subtrees, or even trees, increases the breadth of the search.

Figure 4.6 Complete Trees Represented by One Subtree

How does one measure the fitness of a subtree? Since a subtree is still a tree, fitness can be calculated, as in the SGA, as a function of the gene values. However, if this function is also dependent on the ordering of the genes, and unknown a priori, the TGA and STGA are our only two approaches that will discover the ordering.

How are a chromosome and its parts processed simultaneously? The answer is to first establish a method, or methods, of determining the fitness of complete and incomplete chromosomes (subtrees), and then to rank each one's fitness relative to the size of the fittest known chromosome, be it a whole or a part. When two chromosomes have the same fitness, but have different sizes, the fittest chromosome shall be the larger of the two. In this manner, parts are assembled into larger parts, and the process repeated until the desired chromosome is constructed. However, because all chromosomes and their parts can be ranked relative to one another, they can exist in the same population.

Should the structure of the chromosome be a hierarchy? When the gene dependencies are known a priori, a simple list provides the best structure, as demonstrated by the SGA. However, when the gene dependencies are not known and this structure is used, as in the RGA, the approach fails. The structure to use in this case is a tree, as demonstrated by the TGA; a tree can be used to evolve both the organization and content of the chromosome. However, if fitness values can be determined for the parts of a tree (the subtrees), the best

structure for a chromosome is a subtree that grows into complete trees, as demonstrated by the STGA.

Is there any benefit to evolving the organization of the chromosome? The answer is yes. When gene dependencies are unknown, the simple approach of the RGA, as shown, fails. Since, in most complicated systems, it is an impossible task to determine all gene dependencies, the only way genetic methods will succeed is by determining these dependencies. Since GAs are good at searching, it seems an obvious choice for them to evolve their own organization!

Figure 4.7 Gene Ordering Dependent GAs

4.2 Genome

A SimOrg's complete description is stored in its genome located at its genotypic level. Often this material is simply referred to as a chromosome [Goldberg, 1989] because, for example, in GAs a single chromosome is the only component of the genome. However,

since a SimOrg has more than one component to its genetic information we shall use the term genome to refer to it.

The genotypic level's genome contains two components: a chromosome hierarchy and a motivational drive hierarchy. The chromosome hierarchy contains parameters defining a SimOrg's structures and its control parameters: for example, its memory capacity and its ability to locate plans in short-term memory. The motivation drive hierarchy defines the precedence of a SimOrg's drives.

In GAs, a chromosome is represented as a sequential bit string. In the last section we discussed the problem of using sequential structures for chromosomes. Is there any benefit in binary genes? The reason that a GA uses binary genes stems from the principle of minimal alphabets. This principle states:

'The user should select the smallest alphabet that permits a natural expression of the problem.' [Goldberg, 1989]

This principle suggests that one should use a binary alphabet. The mathematical argument is that the number of schemata available in the binary encoding is greater than the number of schemata in a nonbinary encoding. The implication is a binary alphabet offers the maximum number of schemata per bit of information of any coding. This argument is somewhat misleading. As Antonisse [Antonisse, 1989] has shown, the number of schemata is potentially increased by using a nonbinary alphabet, refuting the principle of minimal alphabets. Although a binary alphabet makes the analysis of a GA and the application of genetic operators easier, a non-binary representation allows GAs to work with more complex representations, and, if the power of a GA's search is related to the number of schemata in a population, then a non-binary representation should enhance a GA's performance.

Why should one use a static chromosome of 0's and 1's? First, there has never been a use for having extra genes. Every gene accounts for some aspect of the phenotype. Second, a static structure of 0's and 1's makes operator computations, such as mutation, crossover, and inversion, easy. Third, it was believed that one could determine the meaningful building blocks a priori. However, the use of sequential, binary chromosomes has its disadvantages: if we want to evolve the organization and contents of the genome's components, a static structure will not work; if we use binary genes, we decrease the number of schemata processed; and, as previously described, it is often impossible to determine meaningful building blocks a priori. Therefore, such simplifications ultimately decrease the flexibility and expressiveness of the evolutionary system.

We prefer to model the genome's components as hierarchies, where genes are not located at fixed positions, but operate as nodes in a hierarchy and are free to move and connect with any other nodes. This enables the genome's components to organize themselves with good gene building blocks. Genes are also permitted to use symbolic values such as integers, booleans, probabilities, and other structures such as drives, rather than 0's and 1's.

In the following section we describe the genome's two hierarchies in more detail.

4.2.1 Chromosome Hierarchy

We believe that it is impossible a priori to determine the exact size of the chromosome and to determine what genes depend on one another and should, therefore, be placed close to one another to form meaningful building blocks.

If the former belief was also held in biology we would expect to see more genes in the chromosome than is actually required. In fact, we would also expect that some of the genetic material is unused, at least until the genes participate in the specification of a phenotypic characteristic. Looking at the biological properties of DNA, this is what one

2

MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS
STANDARD REFERENCE MATERIAL 1010a
(ANSI and ISO TEST CHART No. 2)

finds! The physical characterization of a genome shows that what is present is not all phenotype determining [Doolittle, 1988]. Most of the genes are repetitive sequences or just extra genetic material.

Why are there extra genes? There are two answers to this question. The first answer is that some future combination of genes may be able to create a structure, or behavior, that will benefit an organism. This notion is called *future utility* [Campbell, 1985]. The second answer is that they are there to promote and direct the process of evolution; they are not present for adapting an organism to its environment, but to aid its capacity to evolve. For example, the shuffling of repetitive gene sequences may cause a genome to become unstable - possibly due to an increase in the number and severity of mutations and crossover operations - which can be seen as an attempt to speed up evolution.

Recognizing the problems of fixed size, fixed location, binary chromosomes, we propose the following model for the genome's chromosome hierarchy:

- A conceptually infinite set of genes.
- Genes may use different binary or non-binary alphabets.
- Genes are arranged into a hierarchy; therefore, genes form relationships with one another.
- Genes define both the structure and control parameters of a SimOrg.

This model is a departure from the simple notion of a chromosome presented by the GA community. First, we assume that there is a conceptually infinite set of genes. The reason for this assumption is that we want the system to be able to gracefully handle the addition of new genes to the chromosome hierarchy: once added, it should locate the correct positions and values for them, and it should work with chromosomes of different sizes.

Second, we assume that genes can use different alphabets. The reason for this assumption is to increase the flexibility and expressiveness of the system.

Third, we assume that genes are hierarchically organized. The reason for this assumption is that we do not know what the gene dependencies are a priori, and, by evolving hierarchies similar to the TGA, we want the system to discover them for itself.

Fourth, we assume the genes define the structure and control parameters of a SimOrg, and define the control parameters of evolution within the SimOrg's three levels - genes control the adaptation process! The values of these parameters are not determined by the user, rather, they are determined by the system. We are not proposing that the system create its own parameters, but it will determine the correct values for the ones we give it.

If we can not determine the gene dependencies a priori, how can we determine the system's parameters a priori? GA's often begin with mutation and crossover rates of 0.05 and 0.3, but these rates are then 'twiddled' by the user to get the best performance from the GA. Therefore, determining the parameter values of a GA seems as unjustified as determining its fitness function. Since a GA is good at searching why not have it determine these values? The chromosome hierarchy and its evolution do just that, not just for the mutation and crossover rates, but for all parameters!

The following is a list of the chromosome hierarchy's genes adapted from [Coderre, 1987; Travers, 1987; Beer & Chiel, 1991 Tyrrell & Mayhew, 1991; Schnepf, 1991]. They are initially configured in a binary tree, and later, as a result of the genome's crossover with a mate's genome, form an n -ary tree :

- **MutationRate:** The mutation rate of a SimOrg's genome hierarchies and knowledge; (Probability).
- **CrossoverRate:** The crossover rate of a SimOrg's genome hierarchies and knowledge; (Probability).

- **LongTermMemorySize:** The schema capacity of long-term memory; (Integer).
- **ShortTermMemorySize:** The context capacity of short-term memory; (Integer).
- **TouchSensitivity:** The sensitivity of a SimOrg's touch; (Real Value Range [0..1]).
- **TasteSensitivity:** The sensitivity of a SimOrg's taste; (Real Value Range [0..1]).
- **SmellSensitivity:** The sensitivity of a SimOrg's smell; (Real Value Range [0..1]).
- **HearingSensitivity:** The sensitivity of a SimOrg's hearing; (Real Value Range [0..1]).
- **SightSensitivity:** The sensitivity of a SimOrg's sight; (Real Value Range [0..1]).
- **ConditionDropping:** The ability for a SimOrg to drop missing conditions when applying schema templates to short-term memory and determining a schema's fitness; (Boolean).
- **StepSize:** The length of a SimOrg's step; (Integer).
- **Difference:** The allowable difference between a schema's positive and negative feedback counters; (Integer).
- **CostOfThinking:** The energy cost of thinking per unit of time; (Integer).
- **LifeSpan:** The life span of a SimOrg; (Integer).
- **SchemataTypes:** The types of schemata a SimOrg can notice in memory: planning, analogy, causality; (Set).
- **NumberOfNovelConcepts:** The number of schemata used in the Evolution-Based Reasoning loop; (Integer).

- **EBRLoopThreshold:** The minimum fitness level that knowledge, produced by Evolution-Based Reasoning, must achieve before it can prematurely stop EBR's reasoning loop; (Integer).
- **KnowledgeThreshold:** The minimum fitness level that knowledge, produced by Evolution-Based Reasoning, must achieve before being allowed as a possible answer to SimOrg's current problem; (Integer).
- **MaximumGenerations:** The maximum number of generations permitted by Evolution-Based Reasoning loop; (Integer).

A new chromosome hierarchy is randomly initialized as follows: 1) a root gene is randomly selected and then two randomly selected genes are linked to it; and, 2) the same process is repeated for all of the newly added genes until no more genes are left to be added.

Since we are using a rooted tree to represent the chromosome hierarchy, and every genome's chromosome hierarchy has the same genes, the chromosome hierarchy is similar to the one used by the TGA described in section 4.1.1.3. Being similar, the genome's chromosome hierarchy crossover operator is identical to the TGA's. Although initially structured as a binary tree, the application of the TGA's crossover operator to the chromosome hierarchy results in an n -ary tree. The benefits of using a rooted tree for the genome's chromosome hierarchy and the TGA's crossover operator are the same as those for the TGA: the ability to evolve both the organization and values of the chromosome hierarchy's genes.

The chromosome hierarchy's mutation operator is similar to the mutation operator used by GAs: every gene is subjected to the mutation operator. The decision to mutate a gene is based on a current mutation rate, which in our case is defined by a corresponding gene in the chromosome hierarchy - mutationRate. Since we are using non-binary alphabets for the

genes the mutation operator must take into consideration the type of gene being mutated. There are three types of genes in the chromosome hierarchy: integer, probability, and boolean. Each integer gene is initialized with a maximum, a minimum and a current value. The mutation operator changes the gene's current value by a certain percentage - defined by the mutation rate - within the acceptable minimum and maximum values. For example, if the gene's maximum, minimum, and current values are: 10, 0, and 5, and the current mutation rate is 10%, the gene would change by plus or minus 1. This helps to constrain the mutations from being too severe. A probability gene is similar to an integer gene, except its minimum and maximum values are always 0 and 1. A boolean gene's mutation results in the negation of its current value. Many other types of genes could be used, provided each defines its own mutation operation.

Since a new chromosome hierarchy starts with only randomly generated gene associations (building blocks), how do good, strong associations form? The evolution of the genome and its chromosome hierarchy works them out. It is the model, not the user, that finds the best associations.

In summary, the chromosome hierarchy, by organizing itself as a hierarchy of multiple gene types of no fixed location, containing the parameters for the rest of the system, is developed to be a highly flexible and adaptable structure.

4.2.2 Motivational Drive Hierarchy

The Motivational Drive Hierarchy organizes a SimOrg's drives and is used to help it determine which behavior it is to engage in next. There are seven drives:

- **Thirst:** physical requirement for water.
- **Hunger:** physical requirement for food.
- **Sex:** physical requirement to reproduce.
- **Existence:** physical requirement to protect oneself.

- **Pleasure:** psychological requirement for pleasure.
- **Pain:** psychological requirement for pleasure.
- **Curiosity:** psychological requirement for knowledge.

Each drive has a current value, a threshold value that indicates at what value the drive becomes activated and a shutOff value that indicates at what value the drive becomes deactivated.

Each drive behaves like a thermostat: external stimuli (such as food) and internal behaviors (such as consuming food) increase or decrease the current value of a drive. If a drive's current value becomes greater than its threshold value, the drive is activated and motivates a SimOrg to decrease that drive. The drive remains active until its current value falls below the shutOff value.

Since many drives may be active at any moment there must be a way to prioritize them. To identify the priorities of the drives, they are arranged into a hierarchy called the Motivational Drive Hierarchy and stored as part of the genome. The highest priority drive is found at the root node of the hierarchy. The closer a drive is located to the root drive, the greater its priority. When there are multiple drives active, the drive with the greatest priority is chosen as the active drive. If there are multiple drives active at the same level, the current drive is chosen randomly.

The exact level at which to place each drive in the hierarchy and the appropriate settings for its minimum and maximum values is not known a priori. Therefore, the evolution of the Motivational Drive Hierarchy must determine them. Since every SimOrg has the same drives, and they are organized into a hierarchy, the evolution of the Motivational Drive Hierarchy will be treated in a manner similar to the genome's chromosome hierarchy, and the TGA's chromosome. Therefore, the mutation and crossover operators for the

Motivational Drive Hierarchy are the same as the chromosome hierarchy's operators, except the mutation of a drive gene must be defined.

The mutation of a drive gene changes the gene's minimum and maximum values. These values are in the range zero to one, with the minimum value being less than the maximum value. After the decision is made to mutate a drive, both the minimum and maximum values are subjected to mutation similar to a probability gene's mutation: they are both changed by a percentage equivalent to the current mutation rate, but constrained by the range zero to one. For example, if the drive's minimum value is 0.5 and the mutation rate is 0.1, the value can change to any value between 0.45 and 0.55. If the minimum value becomes greater than the maximum value, it becomes the new maximum values, and vice versa. The drive always maintains its current value.

Like the chromosome hierarchy, the drives are initially arranged in a binary tree. However, after that generation, as a result of the mutation and crossover operators, the Motivational Drive Hierarchy changes shape as it acquires the correct arrangement of drives and their values for the environment.

4.3 Evolution and Fitness of the Genome

In an environment containing a population of SimOrgs, one also finds a population of genomes - each SimOrg contains one genome. Since a SimOrg's genome is only crossed with another during reproduction, its evolution depends on when SimOrgs mate. Since only the fittest of SimOrgs mate - those that manage to survive - only those genomes that define these fit individuals are used to produce new SimOrgs - there is no explicit genome fitness function. During reproduction, genomes are subjected to the operators of mutation and crossover. The newly constructed genomes then become part of the next genome population inside the SimOrgs they help define. Therefore, because variation, natural

selection, and inheritance are present in the genome population (see chapter 2), the genotypic level is a true, and our first, evolutionary level.

The algorithm for the mating of one SimOrg with another is as follows:

mateWithAnotherSimOrg: aSimOrg

 | newGenome newSimOrg |

 newGenome := self genome mateWithAnotherGenome: (aSimOrg genome).

 newSimOrg := SimOrg newWithGenome: newGenome.

 ^ newSimOrg

The algorithm simply takes the two genomes from the parent SimOrgs, 'self' and 'aSimOrg', and creates a new genome by mating them, using the method mateWithAnotherGenome:. The new genome is then used to instantiate a new SimOrg, which is returned.

The algorithm for the method mateWithAnotherGenome: is as follows:

mateWithAnotherGenome: aGenome

 self mutateChromosomeHierarchyBy: (self mutationRate).

 aGenome mutateChromosomeHierarchyBy: (aGenome mutationRate).

 self mutateMotivationalDriveHierarchyBy: (self mutationRate).

 aGenome mutateMotivationalDriveHierarchyBy: (aGenome mutationRate).

 self chromosomeHierarchy crossoverWith: (aGenome chromosomeHierarchy).

 self motivationalDriveHierarchy crossoverWith:

 (aGenome motivationalDriveHierarchy).

 ^ self

The algorithm first mutates the source genome ('self') and the mate's genome (aGenome) by their respective mutation rates - found in the corresponding gene in their chromosome hierarchies. Then, using the TGA's crossover operator, the genome's two hierarchies are crossed with one another. The source genome now contains the newly crossed hierarchies, and is returned. As mentioned, the selection of which SimOrgs mate with one another is decided at the phenotypic level, and we leave that discussion until chapter 5.

Does the genotypic level evolve? Do SimOrgs improve in future generations as a result of their genomes producing better SimOrgs generation after generation? The answer is yes, and we describe these results in chapter 7.

4.4 Chapter Summary

In summary, the evolutionary process is active at many hierarchically related levels; the genotypic level being the most primitive one. A SimOrg's description is stored at this level. Often simply referred to as a chromosome because of that being its only component, our SimOrg has more than one component to its genetic information. Therefore, we use the term genome to refer to the descriptive information.

In situations in which gene dependency information cannot be determined a priori, a GA that uses a static list chromosome structure will often not work. For example, a GA will not work if it uses a fitness function that requires two specific genes to be located beside one another for optimal fitness, and it does not provide an operator for this to occur in the chromosome, such as inversion¹². The notion that there are situations in which fitness and operators must take into account the organization of the chromosome is not peculiar. For example, in genetic programming [Koza, 1992] the very problem is to

¹² One of the problems with a unary operator such as inversion is that the new chromosome it produces does not gain from the experiences of other chromosomes.

determining the correct gene organization. There are situations, however, where GAs work even though their gene dependencies cannot be determined a priori [Grefenstette, 1989; Gabbert *et al.*, 1991]. These successes may be due to a lack of dependencies or to the fact that the encoding happened to reflect them.

Since we do not know a priori the exact organization and contents of the genome, because details of the selection process are unknown, and since we prefer to use a rich symbolic representation that gives us greater flexibility and increases the number of schemata processed, we structure it using hierarchies: a chromosome hierarchy and a motivational drive hierarchy. Our results from the TGA and the STGA demonstrate that hierarchies can be used to evolve both organization and contents.

The chromosome hierarchy contains parameters defining a SimOrg's structures and its control parameters, e.g., its memory capacity, its life span, the mutation rate, the sensitivity of its sensors, and its motivational drive priorities. Recognizing the problems of fixed size, fixed location, binary chromosomes, we propose the following chromosome hierarchy model: a conceptually infinite set of genes, each with a different binary or non-binary alphabets; genes are arranged into a hierarchy, therefore, they form relationships with other genes; genes define the structure and control of a SimOrg; and they are used to control evolution within the genome. The chromosome hierarchy's mutation and crossover operators are the same as those for the TGA, except its mutation operator uses different methods for mutating its three types of genes: integer, probability, and boolean.

The Motivation Drive Hierarchy organizes and defines the priorities of a SimOrg's drives. Each drive has a current value, a threshold value that indicates at what value the drive becomes activated, and a shutOff value that indicates at what value the drive becomes deactivated. The mutation and crossover operators for the Motivational Drive Hierarchy are also the same as the TGA's, except the mutation of a drive gene is newly defined.

The genome and its two hierarchies define the structure and control of a SimOrg, and the control of evolution within its three levels - genes control the adaptation process! Every parameter used by the system is set by this level. This is the start of a parameterless system. We are not proposing that the system will create its own parameters, but it will provide the values for the ones we give it.

Chapter 5

The Phenotypic Level

When one thinks of evolution one naturally considers the evolution of entities, such as beings and organisms, rather than the evolution of genomes and genes or knowledge. When evolution is operating on SimOrgs rather than on genomes, CompEE's view is focused on a SimOrg's phenotypic level.

As described in the last chapter, the evolution of genomes at the genotypic level determines the 'best' data for instantiating SimOrgs. At the phenotypic level, once instantiated, a SimOrg develops and demonstrates its observable characteristics (traits) and behaviors by interacting with its environment. Some of them die because of their poor performance in the environment, while other stronger ones live to reproduce. By dying or reproducing, a SimOrg is affecting the genome populations of future generations - only strong ones have their genomes inherited by the next generation .

As is the case with GAs, a phenotype is represented as a single value computed by a predefined fitness function that uses a phenotype's corresponding genotype - a static structure of characteristics - as input. One of the problems for those using GAs is determining a fitness function that makes a fair comparison of genotypes, i.e., one that provides a continuum of fitness values, not one that simply ranks genotypes as good or bad. By contrast, at the phenotypic level there is no explicit fitness function; fitness is measured simply by the fact that a SimOrg survives in its environment and manages to

reproduce. Knowing that a SimOrg is a nonlinear function of its genome [Langton, 1989], and that its resulting characteristics and behaviors are nonlinear functions of one another, it seems unreasonable to model a SimOrg as a simple fitness value. Rather, a more flexible representation is required for it, one that permits causal relationships to exist between its characteristics, and one that provides for the expansion of new characteristics - phenotypic learning.

In addition to these requirements, we want a SimOrg to be able to do the following:

- 1) Learn regularities about the world.
- 2) Detect properties of the world that can directly affect its characteristics and ultimately its genome.
- 3) Learn which observed changes can be ignored in general or for certain purposes (not all detectable properties are relevant).

In other words, we want our system to be able to do more than just species learning; we want each SimOrg to be able to learn its own reliable, useful information - individual learning.

With these requirements in mind, we have chosen to construct a SimOrg out of four simple systems: a Motivational System, a Stimulus System, a Memory System and a Problem-Solving System. Although additional systems could be defined, these provide the capacity for a cognitive level at which individual learning occurs. Their development is the concern of a SimOrg's phenotypic level.

In this chapter, we examine these four systems in detail and describe how they combine to control a SimOrg in its environment. We also discuss how it reproduces, and how its development and operation in the environment lead to its evolution. The final section summarizes the chapter.

5.1 Motivational System

The Motivational System represents a SimOrg's internal state and helps it to select a goal to pursue. The internal state is represented as a hierarchical arrangement of drives, called the Motivational Drive Hierarchy, whose organization is defined in the genome (section 4.2.2). A SimOrg has seven drives: thirst, hunger, sex, existence, pleasure, pain, and curiosity. The activation of a drive corresponds to strengthening of the motivation for a SimOrg to achieve a particular goal: the goal of finding water, the goal of finding food, the goal of reproduction, the goal of protecting itself, the goal of obtaining pleasure, the goal of decreasing pain, and the goal of seeking knowledge. The selection of which goal to pursue is based on which drives are active and their position in the Motivational Drive Hierarchy.

A drive is based on a model of physiological homeostasis¹³. It has a current value, an activation value that indicates the range of values within which the drive becomes activated (e.g., 0.8 - 1.0), and a deactivation value that indicates the range of values within which the drive becomes deactivated (e.g., 0.0 - 0.35).

Figure 5.1 A Drive

¹³ The principle that, when a physiological variable departs from a set-value, action is taken to correct the deviation.

External stimuli (such as food) and a SimOrg's actions (such as the consumption of food) will either increase or decrease a drive's current value - different stimuli and actions affect each drive differently (described in section 3.4). If a drive's current value becomes greater than its activation value, the drive becomes activated and starts motivating a SimOrg to decrease that drive. For example, if a SimOrg moves about its environment, the current value of its hunger drive will slowly increase, eventually activating its hunger drive and directing it to satisfy the goal by locating food. However, if it manages to eat an object, and that object is food, the current value of its hunger drive will decrease, which, if the value is less than the deactivation value, will deactivate the drive.

The next goal for a SimOrg to pursue is simply the Motivational System's currently active drive. However, since many drives may be activated at the same time, there must be a way to prioritize them so that the Motivational System can select one goal to return - the goal of the highest priority, active drive. To identify the priorities of the drives, they are arranged into a rooted tree called the Motivational Drive Hierarchy. The priority of a drive is represented by its level in the Motivational Drive Hierarchy. The highest priority drive is found at the root node of the tree; the closer a drive is located to the root drive, the greater its priority; and drives located at the same level in the tree have the same priority. When there are multiple drives active, the drive with the greatest priority is chosen as the active drive. If there are multiple drives active at the same level, the current drive is randomly chosen from them.

There are many different models of motivation [Toates, 1986]. These models often differ because each address different features of the environment or an organism's behavior. However, they all agree on the idea that it is both internal and external factors that contributed to the current motivation state. This is also true of our model. Different features of the environment increase or decrease a drive's current value, as will different actions performed by a SimOrg. These changes can result in the drive's activation or deactivation.

However, a drive's activation level does not determine alone the next goal for a SimOrg to pursue. The arrangement of the active drives in the Motivational Drive Hierarchy helps to make the final decision.

5.2 Stimulus System

The Stimulus System has control over a SimOrg's five senses of sight, smell, touch, taste, and hearing. Its responsibility is to return stimuli conjunctions that represent what a SimOrg's five senses can detect from objects at its current position in the environment.

Each sense is assigned a sensitivity value that is stored in a corresponding gene in the chromosome hierarchy found in the genome. For sight, smell, and hearing, the sensitivity value refers to how far away the sense can detect a stimulus. For taste and touch, the sensitivity value refers to the amount of stimulus that must be present for the sense to detect it.

When a SimOrg senses an environmental object it forms a conjunction of features that it can detect about the object. Table 7.2 describes the features of every type of object that can be sensed in the environment. For example, a SimOrg might sense another one as having color white, feeling soft, smelling good, tasting sour, and sounding loud. However, these features would only be detected if all of its five senses were operational, and the amount or proximity of the stimuli was enough to trigger each of the five senses.

The conjunction a SimOrg forms for one object may be the same as the conjunctions it forms for other objects. For example, if a SimOrg that can only smell, touch, and see, came across another one, it would form the conjunction (smells(good), feels(soft), looks(white)), which is the same as the conjunction it would form for delicious food. However, in situations where not all of the features overlap, a SimOrg should be able to determine the discriminating features of the objects and use only those when problem

solving, i.e., when determining what to eat. Since the task of learning discriminating features is part of the cognitive level we leave its discussion until later.

5.3 Memory System

The Memory System has control over a SimOrg's two types of memories: long-term memory and short-term memory. The purpose of short-term memory is to store perceived information. However, this information will quickly decay unless its is rehearsed or repeated. The purpose of long-term memory is to store information that a SimOrg wants to remember but is not currently using. This information can also be lost if it is not useful to the SimOrg.

Short-term memory consists of a fixed size list of contexts which are formed from a SimOrg's sensations and its active drive, each followed by its response action. As described in section 5.2, the sensations are feature conjunctions returned by the Stimulus System. The active drive represents the currently pursued goal.

Short-term memory is simply a list of context-action pairs - the action performed either in response to its current sensations, the active drive, or both. For example, using A as action, C as context, and t as time, the fixed size short-term memory list would appear as follows:

$$A_t C_t A_{t-1} C_{t-1} A_{t-2} C_{t-2} \dots\dots\dots$$

In short-term memory, recent actions and contexts are placed at the head of the list, and, if it becomes full, space is made for new actions and contexts by discarding those at the tail of it.

Long-term memory consists of a fixed size list of schemata. A schema is a tripartite structure consisting of a conditional context, an action, and an expected context, as shown in figure 5.2.

Figure 5.2 A Schema

A schema is used to represent both episodic and semantic information, i.e., information about specific events or episodes which occur in a particular place at a particular time, and de-contextualized facts about the entities and relations between entities in the environment. To represent these types of information, a SimOrg has four types of schemata: a plan or hypothesis schema, an analogy schema, a concept schema, and a complex schema. Each one is described in detail in section 5.3.1.

Schemata are created by two different types of methods: by the application of schema templates to short-term memory looking for repetitive patterns of sensations and actions; and by the Problem-Solving System, described in section 5.4.

The first method is based on four assumptions: 1) that SimOrgs have 'built-in' schema templates for forming schemata; 2) these templates are themselves a result of evolution and their presence is controlled by the genome; 3) that such templates are genetically hard-wired and imposed on nature to break the unending sequence of events into law-like chunks; and, 4) that the majority of template instantiations are reasonable and generally trustworthy.

The construction and application of the templates are based on adaptations to some of Mill's methods of experimental inquiry aimed at discovery of causes [Mill, 1911; Skyrms,

1975], and Riedl's principles of the way organisms extract life-preserving information from their environments [Riedl, 1980].

Although Mill presented five methods for experimental inquiry, we are only interested in his first three methods:

- *Mill's Method of Agreement (MMA)*: 'If two or more instances of the phenomenon under investigation have only one circumstance in common, the circumstance in which alone all instances agree is the cause (or effect) of the given phenomenon' [Mill, 1911].
- *Mill's Method of Difference (MMD)*: 'If an instance in which the phenomenon under investigation occurs, and an instance in which it does not occur, have every circumstance in common save one, that one occurring only in the former; the circumstance in which alone the two instances differ is the effect, or the cause, or an indispensable part of the cause of the phenomenon' [Mill, 1911].
- *Mill's Joint Method of Agreement and Difference (MJMAD)*: 'If two or more instances in which the phenomenon occurs have only one circumstance in common, while two or more instances in which it does not occur have nothing in common save the absence of that circumstance, the circumstance in which alone the two sets of instances differ is the effect, or the cause, or an indispensable part of the cause of the phenomenon' [Mill, 1911].

Riedl was interested in the precise nature of the innate dispositions that controlled living system's behavioral capacities. He proposed a system of four principles that represent an

organism's behavioral constraints. The first principle (RP1) is 'the principle of the re-establishing the established'. It states that the probability of any particular event happening increases with the number of confirmed expectations. The second principle (RP2) is based on 'ignorance of the dissimilar in the similar'. It states that, in looking at similar events one can disregard the differences, and is similar to Mill's method of agreement. The third principle (RP3) states that the series of states and events in nature is not usually arbitrary. In other words, 'increasing constancy of conjunction between events, increases the probability that the events are causally related, and the former event is the cause of the latter'. The last principle (RP4) states that two or more objects, having several features in common, will serve the same purpose. To Riedl, these innate dispositions, or ratiomorphic apparatus, are not conscious expectations, rather they function in a way analogous to consciousness. As Wuketits puts it: 'they operate as a logic of life' [Wuketits, 1986].

Based on the above principles and methods, we use four variations of them for the construction of schemata from short-term memory. The first one, called the principle of presumption of redundancy and regularity, states what ever happens, expect it to happen again. The second one, called the principle of enumerative induction, states the more often something occurs, the more strongly it is expected. The difference between these two is the former one states one should expect future events to happen in the same order as they were seen in the past, and the latter states this expectation should become stronger the more often it occurs. The third one, called the principle of simple analogy, states two different objects used for the same purpose can also be used interchangeably for a different purpose. The fourth one, called the principle of simple concepts, states if two or more features occur together in many confirmed cases, then infer that they are related to one another.

All of these principles use a method called condition dropping to weed out incidentals and irrelevancies. It simply allows features in two contexts that do not match to be dropped from further consideration.

The way we apply the built-in templates, based on the above principles, to construct schemata from short-term memory's list of actions and contexts is described in the next section. A schema constructed from a template is placed in long-term memory for use by the problem solving system. It is in long-term memory that the reliability of the schema is put to the test. However, because templates are reasonable and trustworthy so are most of the schemata they produce, but not all. Good schemata form the building-blocks for more complex ones; the faulty ones are simply forgotten.

As mentioned, schemata are also created by the Problem-Solving System. We, therefore, leave the discussion of how that system creates schemata to section 5.4.

5.3.1 Schema Types

The only unit of knowledge in long-term memory is a schema. A schema is a tripartite structure, similar to Drescher's schema [Drescher, 1991], consisting of a conditional context, an action, and an expected context. An action is an event that can affect the state of a SimOrg and its environment. A context is a collection of propositions (in the form of conjunctions) about the state of the world, that includes a SimOrg's sensations and its active drive.

A schema is the primary building block for complex knowledge and is interpreted both declaratively and procedurally. Declaratively, a schema makes factual assertions about the environment. Procedurally, a schema states how a decrease in a SimOrg's drive can be achieved.

There are four types of schemata: plan or hypothesis, analogy, concept and complex. The first three simple schemata are created by applying built-in schema templates, based on the discovery principles from section 5.3, to short-term memory .

A plan or hypothesis schema is formed using a template based on the principles of *presumption of redundancy and regularity* and *enumerative induction*. For example, if in short-term memory the sequence:

$$C_t A_{t-1} C_{t-1} \dots\dots\dots C_{t-x} A_{t-x-1} C_{t-x-1}$$

is found, where $C_t = C_{t-x}$, $A_{t-1} = A_{t-x-1}$, and $C_{t-1} = C_{t-x-1}$, applying the template would create a plan schema with the following structure: $C_t A_{t-1} C_{t-1}$. If a SimOrg is able to drop conditions (controlled by its genome) the new context includes only those features that match in the source and destination contexts. A plan schema asserts that if the action is taken when its source context's conditions are satisfied, then the new expected context will be obtained.

An analogy schema is formed using a template based on the principle of *simple analogy*. For example, if in short-term memory the sequence:

$$C_1 A_2 C_2 \dots\dots\dots C_3 A_2 C_2$$

is found, applying the template would create an analogy schema with the following structure: $C_1 \leftrightarrow C_3$. An analogy schema asserts that the two contexts can be interchanged with one another.

A concept schema is formed using a template based on the principle of *simple concepts*. For example, if in short-term memory the sequence:

$$C_1 \dots\dots\dots C_2$$

is found, where C_1 and C_2 share many common features, a concept schema with these common features is created. A concept schema implies that its component features are causally related to one another.

Complex schemata are formed by combining other schemata (complex or simple) into schemata hierarchies. For example, using the simple plan schema in figure 5.1, we wish to build a schema that can be used given the current environmental context C_c and desired expected environmental context C_e .

Obviously the schema in figure 5.3 will not solve the problem. However, if we find the plan schema ($C_c \rightarrow A_2 \rightarrow C_1$) in long-term memory and add it to the initial schema, the resulting one, shown in figure 5.4, is closer to solving the problem, but still does not have the desired expected context.

Then, if we find and add the analogy schema ($C_2 \leftrightarrow C_e$), the resulting schema, shown in figure 5.5, is a solution to the problem. This complex schema declares that to get from context C_c to C_e two actions must be performed: A_2 and A_1 , and that when the context C_2 is reached it can be considered the same as context C_e . After each action there is an expectation of what environmental context will be present, and because each expected context can be verified with the real context when the schema is invoked, it is simple to check if the complex schema is making good predictions about its behavior.

Figure 5.3 Plan Schema

Figure 5.4 Complex Schema **Figure 5.5 Final Schema**

What process builds these complex schemata? The answer to this question is the topic of the next section.

Each SimOrg has the potential to use any one of its defined schemata templates. However, the ability to apply them to short-term memory is controlled by the genome. The chromosome hierarchy contains three boolean genes that control a SimOrg's capability of applying the corresponding template. One does not expect to see SimOrgs that have all three genes turned off. The reason is simple. A SimOrg that does not apply any templates will never have any schemata added to long-term memory. Without them, because a

SimOrg has no information it can use to help it reason about its current situation, it will be forced to always select random actions; by only using random actions, it is sure to die!

5.4 Problem-Solving System

The Problem-Solving System has the responsibility of determining how to meet the SimOrg's current needs. Using the schemata in long-term memory, and the current context, i.e., the current perceived external environment and the active drive, the Problem-Solving System attempts to locate a schema whose action, once performed by a SimOrg, will decrease its active drive.

The Problem-Solving System's main execution sequence is as follows:

solveProblem: activeDrive given: featureConjunctions

| response |

" If the previously activated schema has not been completely used -

complex schemata have more than one action associated

with them - check to see if it can be of use now."

response := self evolveActionFromLastUsedSchema: activeDrive

given: featureConjunctions.

(response isNil)

ifTrue: [

" If last complex schema was of no use, try and locate or evolve a new one

to use from long-term memory."

response := self evolutionBasedReasoningForProblem: featureConjunctions

given: activeDrive.


```
(response isNil)
ifTrue: [
    " If no complex or evolved schemata are of use,
    construct a random one."
    response := self randomAction: featureConjunctions
                                given: activeDrive]].

"Return the schema and its associated action for use by the SimOrg to
decrease its active drive."
^ response
```

Since complex schemata may have more than one action associated with them, and each iteration of the Problem-Solving System results in the execution of one action, the Problem-Solving System first tries to work from its previously activated schema, as shown by the underlined code in the above execution sequence. If the schema is still applicable, its next action is used. The reason for executing only one of a complex schema's actions every iteration is to enhance its testability. Rather than testing whether the entire complex schema succeeds or fails, its components can be verified independently by verifying that each internal schema's expected context matches the real one. In doing this, the failure of a complex schema is determined faster, stopping its execution before it does more harm to a SimOrg.

If the last activated schema is of no use, existing schemata in long-term memory are then used by the Problem-Solving System in two ways. First, if an existing schema exactly matches the current perceived external environment and active drive, its action is returned for the SimOrg to perform as a response to its current context. Second, if no schemata match

exactly, they are used as the raw material from which a solution is built. This solution, a complex or simple schema, may in turn be a building block in future solutions. For example, if a SimOrg is hungry and there is food beside it, and it has an appropriate schema in long-term memory linking hunger, food, and eating, its Problem-Solving System should return the schema's action of eating the food as a solution to its current context. If no such schema exists, the Problem-Solving System attempts to construct one by recombining pieces of existing ones that fit the current context.

If the Problem-Solving System cannot find or produce a suitable schema for the current context, a new one is created with the current context and a randomly selected action. Therefore, when a SimOrg does not know what to do, it at least does something. Even a random action might be correct. The current situation and action are recorded in short-term memory and form the raw data for future schemata template learning.

The reason for using a randomly produced schema when no suitable evolved one can be produced, rather than simply relying on the unsuitable evolved one, is the continual reliance on evolved schemata, when failures occur, may cause the SimOrg to get stuck in a type of infinite thinking loop. If the evolved schema is unsuitable and its execution fails to achieve the desired goal, any other future evolved schemata will likely have the same problem because they are all based on the same schemata in long-term memory. The only way of stopping this loop is by repeatedly applying genetic operators to the evolved schema so as to make it considerably different from the one that it is based on. This may be a suitable approach for this situation but in general the evolved schema should not deviate too much from its original form. Since a schema that has had repeated applications of genetic operators is equivalent to a randomly produced one, we might as well use it in the first place.

Since a SimOrg's knowledge is not preprogrammed it must be learned. One such method of learning schemata is by applying schemata templates to short-term memory, as

described in the last section. A second method of learning schemata is through the construction of new schemata. New schemata are formed by a process called Evolution-Based Reasoning. Similar to Case-Based Reasoning, EBR attempts to reuse past knowledge. However, in contrast to CBR, EBR takes learning and adaptation of knowledge as integral components of the reasoning method. The construction of knowledge is seen as the process of differential selection on variation - evolution.

Since EBR forms the major component of the Cognitive Level, we discuss its details and the way it manipulates schemata in chapter 6. In the following section we examine the internal and external views of a schema and describe how its response is determined.

5.4.1 The Different Views of a Schema and its Response

A complex schema - be it a complex plan, a complex concept, or a complex analogy schema - contains a conditional context and expected context, and may contain one or more actions. In section 5.3.1, figures 5.3 - 5.5, we described the construction of a complex plan schema. Internally, this complex plan schema is constructed from two simple plan schemata, and one simple analogy schema. In figure 5.6, we show the external view of this schema; the box in the middle of figure represents the schema's internal, unseen contents. To an external observer or user of this complex plan schema, the only claim made by it is that to go from context C_c to C_e two actions must be performed: A_2 and A_1 .

Figure 5.6 External View of a Complex Plan Schema

The main difference between a complex and a simple schema is that a complex schema maintains an ordered collection of the schemata that form links between its conditional context and expected contexts.

Using this representation for a schema, there is little external difference between simple and complex schemata of the same type. A complex plan schema may, however, contain more than one action. A complex analogy schema appears identical to its simpler version. For example, the simple and complex analogy schemata shown in figure 5.7 both describe how the contexts C_C and C_E can be considered equivalent. However, the complex schema uses two simple analogy schemata to link the conditional context C_C to the expected context C_E . Since the internal portion - the shaded area - of the complex analogy schema is not visible to a user of the schema, the simple and the complex schemata appear the same. In a manner similar to a concept schema, a complex concept schema also appears identical to its simpler version.

There are many benefits to using this representation. First, complex and simple schemata appear the same, and, as a result, can be used interchangeably with one another. Complex schemata can, therefore, be constructed from other simple or complex schemata, which in turn can be constructed from other simple or complex schemata, and so on. Second, both complex and simple schemata can use the same method for determining fitness as described in section 6.2. Finally, as described in chapter 4, because simple and complex schemata can coexist and coevolve with one another, they can evolve and form relationships with one another.

There are a few restrictions that are imposed on the construction of complex schemata. A complex analogy schema can only be constructed from other simple or complex analogy schemata. The same is true for a complex concept schema; it can only be constructed from simple or complex schemata of the same type. It is the complex plan schema that can be

constructed out of any type of simple or complex schemata. Its only requirement is that it contains at least one simple or complex plan schema.

Figure 5.7 Simple and Complex Analogy Schemata

A simple or complex plan schema relates context to action. When an action is requested from a simple plan schema it returns the only one it has. When an action is requested from a complex plan schema it returns the action that follows the last action it returned. Initially, the first action of the first planning schema - simple or complex - in the schemata collection is returned. When all actions have been returned, the schema returns nil.

After the Problem-Solving System evolves or randomly produces a new schema it is reset¹⁴, and its first or only action returned as a solution to the current problem. As

¹⁴ The next action pointer is set to point to the first action in a complex plan schema, or the only action in a simple plan schema.

described in section 5.4, the next iteration of the Problem-Solving System's main execution sequence attempts to use the same schema again if it is a complex schema. If it is a complex plan schema, its next action is returned, and the process repeated. If it is not a complex plan schema, the Problem-Solving System attempts to build a new one.

5.5 Evolution and Fitness of a SimOrg

At the phenotypic level, a SimOrg's behavior results from its operating characteristics defined by its genome. Its behavior results in either its death, from lack of food, water, pain, or destruction by a predator, or its survival. SimOrgs that manage to stay alive will reproduce. Therefore, at the phenotypic level there is no explicit function used to determine fitness; natural selection favors structures and controls that keep SimOrgs alive to reproduce.

A SimOrg's characteristics that define its behavior are initialized in two ways. Its physical structures, e.g., short-term memory size, are initialized at its instantiation and remain unchanged throughout its lifetime, as are some of its cognitive control parameters, such as the ability to locate plans in short-term memory. However, knowledge structures, e.g., where to look for food, are learned during its lifetime. This process of individual learning is done at SimOrg's cognitive level, and is the topic of the next chapter.

Natural selection favors physical structures, cognitive control parameters and knowledge structures that result in good behavior. How are they passed on from one generation to the next? Physical and cognitive characteristics are passed via the genome. However, since SimOrgs don't communicate with one another, any knowledge structures are lost to the next generation. Although there is an obvious benefit to the next generation if it can immediately learn from the experiences of its parents, we achieve a similar effect by passing good definitions of cognitive and physical characteristics to the next generation

through the genome that should enable its members to establish their own similar, even better, knowledge structures.

As SimOrgs develop and interact with their environment, some of them will live long enough to reproduce, others will die. In order to survive, a SimOrg must make use of the characteristics that have been defined for it in its genome. However if recent changes in the genome prove too severe for it to manage, e.g., having a short-term memory size of length one, it will die. With the presence of morphogenesis, natural selection, and reproduction, the developmental level is our second evolutionary level. How does a SimOrg manage to make use of new characteristics to its best advantage? This is the responsibility of cognitive level.

5.6 The Control Sequence

Every SimOrg in the population is given the opportunity, in turn, to perform one action. Its age corresponds to the number of times it has been given this opportunity. When given the opportunity, the following control sequence that incorporates all of the SimOrg's four systems is used to determine its next action:

ControlSequence

| featureConjunctions goal response effort |

"Part-A"

Update the previously activated schema, e.g., if the previous goal or expected context which are stored in short-term memory, was met, reward the schema; otherwise penalize it"

self performFeedBack.

"Part-B"

"Keep track of the amount of time required to execute the main part of the control sequence"

effort := Time millisecondsToRun: {

"Sense the features of the environment"

featureConjunctions := (self stimulusSystem: environment).

"Retrieve the current goal of the currently active drive"

goal := self motivationalSystem currentGoal.

"Reason and return the response to the goal"

response := self problemSolvingSystem solveProblem: goal
given: featureConjunctions.

"Perform the response"

self perform: response.

"Update short-term memory with the feature conjunctions,
the goal, and the action "

self memorySystem addToShortTermMemoryEvents: featureConjunctions
goal: goal
response: response.

"Have the Problem Solving System look for short-term memory
patterns using the available schema templates"

self problemSolvingSystem lookForMemoryPatterns].

"Part-C"

"Increase the hunger drive proportional to the amount of work done in the
main execution sequence"

self motivationalSystem increaseDrive: #hunger

by: (work / (self genome structure: #costOfThinking))

"Part-D"

"Find out if it is time for the entity to die from old age, lack of food, water, or
from being in too much pain"

((self genome structure: #lifeSpan) <= self age) or

((self motivationalSystem node: #hunger) >= 1.0) or

((self motivationalSystem node: #thirst) >= 1.0) or

((self motivationalSystem node: #pain) >= 1.0))

ifTrue: [self die]

The control sequence consist of four parts. In Part-A, the previously activated schema, whose response was used in the last execution of the control sequence, is rewarded if the previous goal or expected context, stored in short-term memory, is met. Otherwise, the schema is penalized.

A schema has two variables that contain the number of times it has been rewarded or penalized. The method performFeedBack rewards a schema by applying the method Reward to it. What follows is the pseudo-code for the Reward method. The method for Penalizing a schema is identical to this method with the references to 'positive' and 'negative' swapped. In the following code, 'self' refers to the schema being rewarded:

Reward

allowableDifference previousSchema |

"Part-A"

allowableDifference := (self associatedEntity genome structure: #difference).

self positive: self positive + 1.

((self positive - self negative) > allowableDifference)

ifTrue: [

 self negative := 1.

 self positive := allowableDifference + 1].

"Part-B"

self parent positive: self positive.

self parent negative: self negative.

"Part-C"

"Also reward previous schema if it was an analogy or a concept schema"

((previousSchema := self previousResponseSchema) isNil not)

ifTrue: [

 (previousSchema isAnalogy) or (previousSchema isConcept)

 ifTrue: [previousSchema incPositive]]

Part-A of the Reward method, after incrementing the schema's positive variable, ensures that it and the negative variable differ by no more than an allowable difference

defined in the genome of the SimOrg that owns this schema. If the difference between the two variables is larger than the allowable difference, the negative variable is reset to 1, and the positive variable is set to the allowable difference plus one.

The reason for having an explicit difference is to help detect invalid schemata rapidly. For example, if a schema was previously rewarded 99 times and penalized only once, but was recently penalized 49 times, it would appear as if the schema is now invalid. However, if one compares the schema's variable counters, it appears as if the schema has only failed one third of the time. By having an explicit difference, the schema's variables reflect only the recent history of its application, not its entire past.

If this schema is a component of a complex schema, part-B of the Reward method updates the complex schema's positive and negative variables with those of the current schema; it has no effect on simple schemata. Remembering from section 5.4.1, that a complex schema looks like a simple schema to an external observer, it, too, has positive and negative counter variables. The reason for updating the 'parent schema' is to have its counters always reflect those of its previously used part - a complex schema is only as good as its previously used part.

As in part-B, if this rewarded schema is a component of a complex schema and the schema before it in the complex schema is a concept or an analogy schema, part-C of the Reward method will update it. The reason for this is, since a concept or an analogy schema is only good if it is used successfully to associate features or to switch contexts of a plan schema, its success can only be determined by the success of the planning schema that uses it.

If the schema was rewarded and it is a new one (i.e., it was constructed by the EBR process, not just retrieved and used straight out of long-term memory) the new schema is added to long-term memory for further consideration (rewarding and penalizing) and for building material for future schemata.

After rewarding or penalizing the previously activated schema, part-B of the Control Sequence invokes all of a SimOrg's development systems to help it determine its next action. First, the current features of the objects that it can sense are determined by the Stimulus System. Next, the Motivational System determines the active drive and the goal. Then, using the current object features and the goal, the Problem Solving System determines an action to solve the goal, which the SimOrg then performs. The features, goal, and action are then added to short-term memory which is then searched for new source schemata by applying the SimOrg's available schema templates. Finally, the amount of time required for this section to execute is recorded for use by part-C of the method.

In part-C of the Control Sequence, a SimOrg's hunger drive is increased proportional to the time it takes to execute the second part. Therefore, the more time taken in part-B, the more likely a SimOrg will die, unless the extra time taken results in an increase in food consumption. What is desirable is a balance between action processing and the end results of performing the action. A SimOrg does not benefit from taking a long time to think, if it does not result in an increase in food; in fact, it is harmful to it.

Part-D of the Control Sequence determines whether a SimOrg should die, either from old age, lack of food or water, or from too much pain. If it dies, it is removed from the environment.

Upon completing this sequence, a SimOrg waits until it is given its next opportunity to interact with the environment. This only happens after every one else has had their turn.

5.7 Chapter Summary

In CompEE, when evolution is operating on SimOrgs rather than on genomes, CompEE's view is focused on the phenotypic level. At the phenotypic level, once instantiated, a SimOrg develops and demonstrates its observable characteristics (traits) and behaviors by interacting with the environment.

At the phenotypic level, we want a SimOrg to obtain regularities about the world, to detect properties about the world that can directly affect its characteristics and ultimately its genome, and to learn which observed changes can be ignored in general or for certain purposes (not all detectable properties are relevant).

With these requirements in mind, we have chosen to construct a SimOrg from four simple systems: a Motivational System, a Stimulus System, a Memory System, and a Problem-Solving System. The Motivational System represents a SimOrg's internal state and helps it to select a goal to pursue. The Stimulus System has control over its five senses of sight, smell, touch, taste, and hearing. The Stimulus System's responsibility is to return stimuli conjunctions that a SimOrg's five senses can detect from objects in the environment.

The Memory System has control over two types of memories: long-term memory and short-term memory. The purpose of short-term memory is to store perceived information. Short-term memory consists of a limited list of actions and contexts formed from a SimOrg's sensations and active drive. The sensations are conjunctions returned by the Stimulus System at the current position in the environment. Long-term memory is also a fixed size list that contains schemata. A schema is a tripartite structure consisting of a conditional context, an action, and an expected context, and is used to represent both episodic and semantic information.

There are four types of schemata: plan or hypothesis, analogy, concept, and complex. From an external point of view, simple and complex schemata have similar structures: a conditional context, an expected context and possibly one or more actions. The benefits of this representation are that complex and simple schemata can be used interchangeably with one another and that they can use the same method for determining fitness

Schemata are created by two different types of methods: by the application of schema templates to short-term memory, based on adaptations to the methods of Mill and Riedl that look for repetitive patterns of sensations and actions; and by the Problem-Solving System.

The Problem-Solving System has the responsibility of determining how to meet the SimOrg's current needs. Using the schemata in long-term memory, and the current context: its current perceived external environment and the active drive, the Problem-Solving System attempts to locate a schema whose action, once performed by a SimOrg, will decrease its active drive.

Each SimOrg in the population is given the opportunity, in turn, to perform one action. Its age corresponds to the number of times it has been given this opportunity. When given the opportunity, a SimOrg invokes a control sequence that incorporates all of its four systems to determine its next action.

At the phenotypic level there is no explicit function used to determine a SimOrg's fitness. As SimOrgs develop and interact with their environment, some of them will live long enough to reproduce, others will die. In order to survive, a SimOrg must make use of the characteristics that have been defined for it in its genome. However, if recent changes in the genome prove too severe for it to manage, it will die. With the presence of morphogenesis, natural selection, and reproduction, the developmental level is our second evolutionary level.

Chapter 6

The Cognitive Level

In the last two chapters, we described how a SimOrg comes to acquire the values for the structure and control parameters defined in its genome - species learning - and how it, once instantiated by those parameter values, sets out to acquire information - schemata - about the environment using innate schema templates - phenotypic learning. In this chapter, we are interested in how a SimOrg uses those source schemata to evolve reliable information and how it reasons with them - individual learning. When evolution is operating on schemata, rather than on genomes or SimOrgs, CompEE's view is focused on the cognitive level.

A SimOrg's knowledge¹⁵ is viewed as a product of its interaction with the environment. Since factors like memory size and cognitive structures play important roles in the growth of knowledge - they are the tools - the ability of a SimOrg to learn has a close relationship with its biology.

Knowledge is acquired in one of two ways. The first way, as described in section 5.3.1, locates repetitive patterns in short-term memory. This primitive form of knowledge acquisition is by observation, associating particular stimuli with responses. This can be a very costly learning method because, if a SimOrg does not encode its repetitive patterns as

¹⁵ Its schemata in long-term memory.

general hypotheses, it must then attempt to record all of its particular experiences which requires a large amount of memory. The increase in the number of available schemata that might possibly fit a SimOrg's current situation will also create additional costs in locating them and deciding upon which one(s) to use in the situation.

The second way adapts existing knowledge to produce new knowledge. The process of adapting old solutions to new problems is not only cognitively plausible [Riesbeck and Schank, 1989; Ross, 1989] but also rests on the well supported assumption of Case-Based Reasoning [Kolodner, 1988], i.e., that small changes to existing knowledge will often result in further useful knowledge. This form of learning has close parallels to evolution. Initially, variations occur in knowledge, either voluntary (the merging of knowledge) or involuntary (errors in memory or in the copying process). Some variations are beneficial, and constitute new 'knowledge', others do not. Natural selection prefers the former, resulting in a reproduction differential in the knowledge.

At the phenotypic level, pattern learning played an important, initial role by developing a SimOrg's first potential solutions to its tissue needs. At the cognitive level, a SimOrg's tissue needs present it with a specific criterion for judging the adequacy of its solutions. For example, hunger forces it to move around to find food. A decrease in its hunger, as a result of finding food and eating it, is a successful outcome to the proposed solution. Initially tentative solutions are proposed, based on the acquired schema. Some of these schemata are discarded immediately by using existing knowledge to rule them out; others are tried out to determine their appropriateness. From these simple schemata, complex schemata are built; from complex schemata, simple schemata (by splitting complex ones apart) and even more complex ones (by joining complex schemata together) are built; the more knowledge a SimOrg has, the more knowledge it can generate.

Why should a cognitive level exist? The cognitive level develops, validates, and reuses or alters a SimOrg's existing knowledge. It enables a SimOrg to find and make use of the regularities in its world and to survive!

In this chapter, we describe how reliable, useful symbolic knowledge, in the form of schemata, can be produced using the process of evolution. This new model of reasoning is called Evolution-Based Reasoning. Similar to CBR, EBR attempts to reuse past knowledge. However, in contrast to CBR, EBR considers learning and adaptation to be integral components of the reasoning method. The construction of knowledge is construed as a process of differential selection on variation, similar to the one that governs the evolution of SimOrgs.

The approaches of CSs, AL and Animats also take adaptation seriously, but must a priori fix their parameters and structures. For example, CS use static structures to construct knowledge at one level of complexity and for only one problem. EBR simultaneously constructs both simple and complex (multi-layered) knowledge for multiple problems such as feeding, drinking and mating. AL, when considering individual learning, has combined the traditional methods of learning studied by the ML community, such as back-propagation. These models, however, do not examine how individual learning could evolve and function in an organism. Most of the Animats learning architectures are effective at trial-and-error learning and lack the ability to perform planning or reasoning. EBR, with its evolved parameter and schemata - hierarchically organized structures - overcomes these problems by using evolution as the central knowledge gaining process, attempting to better adapt a SimOrg to its given environment.

6.1 Evolution-Based Reasoning

In section 5.6, we described a SimOrg's control sequence, summarized by the following steps: update previously used schema, sense the environment, determine the

currently active drive, determine the next action to take in order to decrease the active drive, perform the action, update short-term memory with the current context and action, look in short-term memory for repetitive patterns using schema templates and determine if it is time to die.

As described in section 5.4, the Problem-Solving System has the responsibility of determining which schema, and its corresponding action(s), a SimOrg should take to decrease its active drive. If the last schema was a complex schema, the Problem-Solving will return it again in order that its next action can be used. If the last schema was not complex, or it was and all of its actions have already been taken, the Problem-Solving System uses Evolution-Based Reasoning to find, either in long-term memory or constructed from it, a schema for a SimOrg to use in order to decrease its active drive.

The execution sequence of EBR is as follows:

evolutionBasedReasoningForProblem: featureConjunctions

given: activeDrive

| anticipatedfeatureConjunctions mate1 mate2 generations population

newPopulation maximumGenerations EBRLoopThreshold

numberOfNovelConcepts knowledgeThreshold bestKnowledge |

generations := 0.

anticipatedfeatureConjunctions := Set new.

"Retrieve EBR parameters from genome"

numberOfNovelConcepts := self genome structure: #NumberOfNovelConcepts.

knowledgeThreshold := self genome structure: #KnowledgeThreshold.

EBRLoopThreshold := self genome structure: #EBRLoopThreshold.

maximumGenerations := self genome structure: #maximumGenerations.

"Step A"

"Establish fitness of memory's schemata based on current requirements
of the SimOrg: its current drive, and its current context"

self memory setFitnessMemory: featureConjunctions

and: anticipatedfeatureConjunctions

and: activeDrive.

"Initialize knowledge population with fittest schemata in memory"

population := self memory selectBest: numberOfNovelConcepts.

(population isEmpty)

ifTrue: [^ nil].

bestKnowledge := population bestKnowledge.

"Step B"

"Try and form a schema that will meet the current problem. Loop for
a fixed number of iterations or until a schema meets a fitness threshold"

[(generations < maximumGenerations) &

(bestKnowledge fitness < EBRLoopThreshold)] whileTrue: [

newPopulation := SortedCollection sortBlock:

(MemorySystem schemataSortBlock).

"Using genetic operators to form new schemata"

(size := population size) timesRepeat: [

mate1 := self selectMateFromPopulationUsingRouletteSelection:

population .

```

mate2 := self selectMateFromPopulationUsingRouletteSelection:
    population .
mate1 := self perform: (self randomGeneticOperation)
    with: mate1 with: mate2.
newPopulation add: mate1].

```

```

population := newPopulation.
generations := generations + 1.
bestKnowledge := population strongestSchema.

```

"Step C"

" Choose context focus for next generation and update fitness values based
on the new focus"

```
((generations mod: 2) = 0)
```

```
ifTrue: [
```

```
    "Use the best schema's contexts as the new focus"
```

```
    self memory setFitnessOfMemory:
```

```
        (bestKnowledge featureConjunctions)
```

```
        and: (bestKnowledge anticipatedFeatureConjunctions )
```

```
        and: (bestKnowledge conditionalDrive)]
```

```
ifFalse: [
```

```
    " Change focus back to original problem contexts"
```

```
    self memory setFitnessOfMemory: featureConjunctions
```

```
        and: anticipatedFeatureConjunctions
```

```
        and: activeDrive].
```

"Step D"

" Add fit schemata from long-term memory into the genetically produced
schemata population for future consideration"

population addAll: (self memory selectBest: numberOfNovelConcepts)].

"Step E"

" Rank the fitness of the schemata in the population relative to
the current problem"

population setFitnessOfMemory: featureConjunctions

and: anticipatedFeatureConjunctions

and: activeDrive.

" If the population's best schema meets a minimum fitness level return it as the entity's
solution to its given problem, otherwise nil"

(population bestKnowledge fitness < knowledgeThreshold)

ifTrue: [^ nil]

ifFalse: [^ bestKnowledge response]

The EBR process can be summarized with the following steps: A) retrieve a population of the fittest schemata from long-term memory for solving the current problem; B) using the available genetic operators (described in section 6.3), form a new population of simple and complex schemata by applying the operators to the existing population's schemata; C) choose new focus contexts (described in detail in the following paragraphs) for the next iteration of the process; D); retrieve existing fit schemata from long-term memory that match the new focus contexts and add them to the population of schemata produced by step C; and E) repeat the process with the new population until a schema that solves the problem is found or the maximum number of generations is reached.

In step A, the best (fittest) schemata for the current problem are retrieved from long-term memory; the current problem consists of decreasing the currently active drive and the current feature conjunctions. As described in section 6.2, a schema's fitness value represents the schema's past successes and how well the schema matches the current problem and anticipated contexts. We leave the specifics of the calculation of a schema's fitness until section 6.2. However, an important feature of a schema's fitness is that it is not a static value; in different contexts a schema will have different fitness values. The first part of step A sets the fitness values of all schemata in long-term memory relative to the focus context of the current problem.

In the second part of step A the fittest n schemata from long-term memory are retrieved and used to form EBR's initial schemata population. The number of schemata retrieved is genetically controlled and stored in the genome. The more schemata that are used, the more processing time EBR will take. Since a SimOrg is penalized proportional to the time taken in its main control sequence, e.g. its hunger drive is increased, there is pressure to keep EBR's processing time down. Therefore, there is pressure to keep the number of new schemata retrieved from memory down to a number that lets EBR work efficiently - a SimOrg should not die thinking about hundreds of schemata if working with only two of them would provide life sustaining results.

In the first part of step B the population of schemata are checked to see if one schema's fitness is above a predefined threshold contained in the genome. If a schema has reached this threshold, EBR's task is complete: the schema's action is a sufficient response to the current problem's context.

If, however, no schema's fitness has reached this level, EBR attempts to build one that does. To do this, EBR constructs a new population of schemata by applying genetic operators to the existing population. First, EBR selects two mates (schemata) from the population using roulette selection [Goldberg, 1989]. The application of a random genetic

operator to these mates, described in section 6.3, results in a new schema which is added to the new population. This process is repeated until the number of schemata in the new population equals the old one.

After creating the new population, step C chooses the new focus contexts. The new contexts are used to re-evaluate the fitness of each schema in long-term memory so that new strong ones can be selected and used in the next iteration of step B that will either extend the population's fittest schemata in new directions or the other schemata in the direction of the problem context. The focusing mechanism does this by changing back and forth between using the contexts of the problem and the contexts of the fittest schema in the new population. This mechanism attempts to combine a phase of exploitation with a phase of exploration. During exploitation the EBR process selects schemata to combine with existing ones that will match the current problem's contexts. For example, when the problem contexts C_c and C_e are used in fitness computation, the plan schema $(C_c A_2 C_1)$ and the analogy schema $(C_2 \leftrightarrow C_e)$ in figure 5.5 would be stronger than the plan schema $(C_1 A_1 C_2)$, and are more likely to be selected because they are a better match with the problem contexts. By selecting these schemata to use with the existing population of schemata, EBR is attempting to construct a schema that solves the problem. During exploration the EBR process selects schemata that are directed to extending (e.g., making a complex plan schema by joining two simple plan schemata) the best schema in the population. For example, when the contexts of the fittest schemata are used, suppose C_c and C_1 of the plan schema $(C_c A_2 C_1)$, the plan schema $(C_1 A_1 C_2)$ in figure 5.5 would be stronger than the analogy schema $(C_2 \leftrightarrow C_e)$, and more likely to be selected because it is a better match with the expected context of the fittest schemata. By selecting this schema to use with the existing population of schemata, EBR is attempting to construct a schema that extends the best one in the population. It may be better to explore more and exploit less, or vice-versa. However, by simply changing back and forth, directing EBR's effort to

solving the problem and then expanding on what is already known, the end effect will be the same, although it might be a little slower.

After establishing the new focus contexts, step D adds a fixed number (defined in the genome) of the fittest schemata from long-term memory, relative to the new focus context, to the population forming the next population for the EBR process. Steps B, C, and D are then repeated until a constructed schema reaches the required fitness threshold, or a fixed number of generations, defined in the genome, is reached. The fixed number of generations prevents EBR from indefinitely attempting to build a schema for the current problem.

In step E, after ranking the fitness of each member of the population to the current contexts, the fittest schema produced by steps A, B, C, and D, is once more checked to see if it meets an acceptable fitness threshold, and, if it does, its action is returned as the response to the current problem; if it does not, no action is returned. In this case, the Problem-Solving System will then use a random action as the SimOrg's response.

There is a balance to be found between the number of schemata processed, the time for processing, and the minimum fitness thresholds that determine what schemata are acceptable for returning their responses as solutions to the current problem. The values of these parameters are determined by species learning at the genotypic level. Their correct settings provide a SimOrg with an EBR process that functions properly, verifying and constructing reliable information about the SimOrg's environment.

With the presence of morphogenesis, natural selection, and reproduction of schemata, the cognitive level forms our third and final evolutionary level.

6.2 Schema Fitness

A schema's fitness is required by two different processes: the EBR process for determining which schemata it should use to solve the current problem, and the process of

long-term memory updating for determining which schemata should be removed to make room for the new ones.

When the first phase of the EBR process locates the best (fittest) schemata to use for solving the current problem, a schema's fitness is calculated as follows:

$$\text{fitness} := \frac{\text{relativeFrequencyOfSuccess} + \text{featureMatch} + \text{driveMatch}}{3}$$

$$\text{relativeFrequencyOfSuccess} := \frac{\text{positive}}{\text{positive} + \text{negative}}$$

$$\text{featureMatch} := \frac{\text{MFICC} + \text{MFIAC}}{\text{TFICC} + \text{TFIAC}}$$

A schema fitness value is a combination of three values: how well the schema has worked in the past (*relativeFrequencyOfSuccess*), how well it matches the current contexts (*featureMatch*), and if its action has been used in the past to decrease the active motivational state (*driveMatch*).

A schema's *relativeFrequencyOfSuccess* is the ratio of the number of times it has been used successfully to the total number of times it has been used.

A schema's *featureMatch* begins by adding the number of features in the schema's conditional context that match with the given conditional problem context (MFICC) to the number of features in the schema's expected context that match with the given anticipated problem context (MFIAC). The sum represents the number of schema features that match the problem's contexts. This value is then divided by the sum of the total number of features in the given conditional problem context (TFICC) and the given anticipated context (TFIAC).

A schema's drive match is either a value of zero or 1. If the schema's active drive, stored as part of its conditional context, matches the SimOrg's active drive, the drive match is 1; otherwise, it is zero.

Each of the above three values is less than or equal to one, and each is evenly weighted in the final calculation of a schema's fitness. Therefore, a denominator value of 3 for the fitness calculation produces a schema fitness value between 0 and 1.

A question to ask here is: Is the fitness value of a complex schema calculated differently from the fitness value of a simple schema? The answer is no. Since the external view of a complex schema is identical to that of a simple schema - as described in section 5.4.1, and in particular figure 5.4 - both have conditional and anticipated contexts, and both have positive and negative variables. Even though a complex schema has considerably more information in it, only its external values are used in the calculation of its fitness. Therefore, there is no difference in the way a complex or simple schema's fitness value is calculated; because of this fact, they can coevolve with one another as part of the same population.

A schema does not have a static fitness value. There are two reasons for the dynamic fitness value. First, its fitness value changes every time it is penalized or rewarded. Second, since every problem's contexts are different, how well a schema matches the contexts differs for each problem.

Since long-term memory is of a fixed size, only the best schemata are kept. In order to add new schemata to long-term memory, weak ones must be removed to make room for them. However, because there is no problem context, how is the fitness of a schema determined? In the absence of any contextual information, it is ultimately how well a schema has performed that determines its fitness. Therefore, in this case a schema's fitness is simply calculated as its relative frequency of success. This value alone determines which schemata are removed from long-term memory to make room for the new ones.

6.3 Schema Operators

After the selection of the best schemata for EBR's working population, genetic operators are randomly applied to produce a new population of schemata. There are four genetic operators: mutation, crossover, split and join.

The mutation operator can randomly add, remove, or alter any of the features in a schema's contexts. It can also randomly change a plan schema action as shown in figure 6.1. The probabilistic decision to mutate a feature or action is based on the SimOrg's mutation rate stored in its genome. In the case of a complex plan schema, the mutation operator is applied to any of its actions, but it is only applied to its externally viewed contexts, as shown in figure 6.2. The reason that the mutation operator is not permitted to alter a complex schema's internal contexts is that we chose to operate on a schema as a whole rather than a collection of parts. Once formed, the important parts of a complex schema are its externally viewed contexts and actions; they are the parts that interact with other schemata. By operating on only these external parts, the operators treat simple and complex schemata the same.

Figure 6.1 Mutation Operator Applied to a Simple Schema

Figure 6.2 Mutation Operator Applied to a Complex Schema

The crossover operator combines different parts (actions, active drive, and features) from two source schemata into one new schema, as shown in figures 6.3 and 6.4 for the cases of simple and complex plan schemata. In the case of a simple plan schema, a fair coin is tossed to decide if the source schema's action and active drive are included in the new schema. A fair coin is then also used to decide which contexts' features are also to be part of the new schema. In the case of a complex planning schema, one point crossover is applied to the actions of the two source schemata to determine the actions of the new schema. The features and active drive of the new complex schema are determined in the same way as those for a simple schema.

Figure 6.3 Crossover Operator Applied to a Simple Schema

As with the mutation operator, and for the same reasons, the crossover operator is not permitted to alter a complex schema's internal contexts, only its actions.

Figure 6.4 Crossover Operator Applied to a Complex Schema

The split operator either breaks one complex schema into two smaller schemata, as shown in figure 6.6; or it splits the contexts of a simple schema in two, forming two simple schemata with smaller contexts, as shown in figure 6.5. In the latter case, a fair coin is used to decide which new schema will contain a particular feature of the original schema. In the former case, a link is randomly selected to break the complex schema.

Figure 6.5 Split Operator Applied to a Simple Schema

The join operator joins two schemata together, simple or complex, forming a new complex schemata.

Although the above examples have mostly demonstrated the application of genetic operators to plan schemata, the operators work the same when applied to concept or analogy schemata. The only difference is that there is no action involved. By taking each of

the above figures, the corresponding ones for the other types of schemata can be produced by simply removing the action notation from them.

Figure 6.6 Split Operator Applied to a Complex Schema

Figure 6.7 Join Operator Applied to a Simple and a Complex Schema

6.4 Chapter Summary

In this chapter we are interested in how a SimOrg uses its source schemata, found in long-term memory, to evolve reliable information about the environment and to reason - individual learning. When evolution is operating on schemata, rather than on genomes or SimOrgs, CompEE's view is focused on the cognitive level. The cognitive level develops, validates, and reuses or alters a SimOrg's existing knowledge.

One way a SimOrg acquires knowledge is by observing repetitive patterns in short-term memory. Another way is by adapting its existing knowledge using a reasoning process called Evolution-Based Reasoning. Similar to CBR, EBR attempts to reuse past knowledge. However, in contrast to CBR, EBR makes learning and adaptation integral components of the reasoning method. The construction of knowledge is construed as a process of differential selection on variation, similar to the one that governs the evolution of SimOrgs.

The EBR process can be summarized as the following steps: A) retrieve a population of the fittest schemata from long-term memory for solving the current problem; B) using the genetic operators of mutation, crossover, split, and join, form a new population of simple and complex schemata by applying them to the existing population's schemata; C) choose new focus contexts for the next iteration of the process; D); retrieve from long-term memory existing fit schema that match the new focus contexts and add them to the population of schemata produced by step C; and E) repeat the process with the new population until a schema that solves the problem is found or the maximum number of generations allowable is reached.

With the presence of morphogenesis, natural selection, and reproduction of schemata, the cognitive level forms our third and final evolutionary level.

Chapter 7

Experiments

Although there are numerous experiments one could conduct with our model, in this chapter we describe the results of those experiments that help to evaluate its key features: the ability of evolution to evolve stable, hierarchical structures and their values, such as the motivational drive hierarchy, the chromosome hierarchy, and schemata; the ability of a SimOrg to learn information about the environment at multiple levels; and the EBR process.

While some experiments demonstrate the model's ongoing acquisition of knowledge at multiple levels, others demonstrate the effects of restricting some of the model's parameters; for example, permanently enabling genes that control a SimOrg's ability to find schemata in short-term memory using schema templates, and not permitting SimOrgs to die.

In the following sections we describe the results from experiments conducted in three different environments: a plentiful environment, a spacious environment, and a hazardous environment. Unless otherwise noted, we report on the median results achieved from running an experiment 10 times. In each experiment, we initialize a population of 10 SimOrgs and 1 predator, each of which uses a genome that is randomly initialized from the conditions shown in table 7.1. For each experiment, the system is then run for a fixed number of deaths (750). In our system, one iteration consists of each SimOrg population

member sensing its environment and performing one action. Every 30 iterations, accumulated statistics are dumped to a file and then reset.

Gene Name	Min - Max
GenomeMutationRate	0.05 - 0.2
GenomeCrossoverRate	0.1 - 0.3
LongTermMemorySize	10 - 30
ShortTermMemorySize	10 - 20
HearingSensitivity	0.0 - 1.0
TouchSensitivity, SmellSensitivity	0.0 - 1.0
SightSensitivity, TasteSensitivity	0.0 - 1.0
Analogy	true-false
Concept	true-false
Planning	true-false
ConditionDropping	true-false
StepSize	1 - 1
CostOfThinking	10000000 - 25000000
LifeSpan	30 - 70
MatingAge	10 - 20
Difference	5 - 20
NumberOfNovelConcepts	3 - 10
EBRLoopThreshold	0.7 - 0.9
EBRKnowledgeThreshold	0.5 - 0.7
MaximumGenerations	2 - 10

Table 7.1 Gene Ranges

7.1 The Environment

A SimOrg's environment consists of a square grid of cells. The middle cell of the grid is located at the position (0,0). SimOrgs or other environmental objects can position themselves on any cell, and several objects on the same cell are permitted.

There are nine different kinds of environmental objects: SimOrgs, Predators, Rocks, Water, Holes, Nonpoisonous Food, Delicious Food, Moldy Food, and Poisonous Food. Each object has the following features: color, smell, taste, sound, and texture, which correspond to the features that a fully functional SimOrg could sense. The following table identifies the different features of each environmental object:

	Color	Texture	Smell	Taste	Sound
SimOrg	white	soft	good	sour	loud
Predator	green	warm	bad	sour	quiet
Water	blue	cold	noSmell	noTaste	loud
Rock	gray	cold	noSmell	sour	none
Hole	black	none	noSmell	noTaste	none
PoisonousFood	blue	cold	bad	sour	none
Moldy Food	green	cold	bad	sour	none
DeliciousFood	white	soft	good	sweet	none
NonPoisonousFood	green	cold	bad	sweet	none

Table 7.2 Environmental Object Features

One important point about the environment is that an environmental object's features overlap with other objects. This means that a SimOrg will have to sort out what features are representative of a particular object.

Another important feature of this environment is that a SimOrg's optimal action can't always be determined immediately from its sensor information. The reason for this is that objects in the environment have common features, e.g. poisonous and nonpoisonous food are both cold to the touch, and, depending on what senses are available to a SimOrg, different objects can be consistent with its stimulation. The result is that a SimOrg's behavior reinforcement is not always immediate. In terms of Wilson's environment taxonomy [Wilson, 1991], our environment is a Class 2 environment, the most difficult of his three classes of environments. In a class 0 environment, there exists at least one action which if taken will result in positive reinforcement for every sensor stimulus. In a class 1 environment, for only some sensory stimuli does there exist at least one action which will result in positive reinforcement.

A SimOrg's actions include the following:

- 1) Drink: Drink an object.
- 2) Eat: Eat an object.
- 3) Sniff: Smell an object.
- 4) Reproduce: Reproduce with an object.
- 5) Movement: Move north, south, east, west, or remain stationary.

A SimOrg and Predator can perform any one of its actions with, or on, each of the environmental objects. By performing an action, it either increases, decreases, or has no effect on any one, or all, of their drives. The following tables - tables 7.3 to 7.7 - show how performing a particular action on an object will affect a SimOrg's or Predator's drives. The way a drive changes depends on both the action and the object of the action. In the following tables, 'I' indicates an increase in the drive, 'D' a decrease, and a blank indicates no change. If 'I' or 'D' is in brackets, it means the predator's drive is affected differently

by the action. For example, table 7.3 shows that a SimOrg's movement away from any environmental object other than a Predator will cause a decrease in its curiosity drive.

Movement	Thirst	Hunger	Sex	Existence	Pleasure	Pain	Curiosity
SimOrg:	I	(I)	(I)	D (I)	I	D ()	D
Predator:	I	(I)	D (I)	I	(D)	I ()	I
Rock:	I	(I)		I		I	D
Hole:	I	(I)	(D)	D	D ()	I	D
Water:	D (I)	(I)		D	D ()	D	D
Del-Food:	I	(I)		D	(I)	D ()	D
NP-Food:	I	(I)		D	(I)	D ()	D
P-Food:	I	(I)		I	(I)	I ()	D
Bad-Food:	I	(I)		I	(I)	I ()	D

Table 7.3 Movement Feedback

Drinking	Thirst	Hunger	Sex	Existence	Pleasure	Pain	Curiosity
SimOrg:	I				I	I	I
Predator:	I		D	I		I	D
Rock:	I					I	I
Hole:	I					D	I
Water:	D				D		I
Del-Food:	I						I
NP-Food:	I						I
P-Food:	I						I
Bad-Food:	I						I

Table 7.4 Drinking Feedback

Eating	Thirst	Hunger	Sex	Existence	Pleasure	Pain	Curiosity
SimOrg:	I ()	I (D)			I	I ()	I
Predator:	I ()	I	D	I		I	D
Rock:	I ()	I				I	I
Hole:		I					I
Water:	D					I	I
Del-Food:		D			D		I
NP-Food:		D			D		I
P-Food:		D			I	I	I
Bad-Food:		D			I	I	I

Table 7.5 Eating Feedback

Sniffing	Thirst	Hunger	Sex	Existence	Pleasure	Pain	Curiosity
SimOrg:	I	I	(D)	(I)	D ()	(D)	I (D)
Predator:	I	I	D ()	I ()	(D)	I ()	D (I)
Rock:					I		I
Hole:					I		I
Water:					I		I
Del-Food:					I	D	I
NP-Food:					I	D	I
P-Food:					I		I
Bad-Food:					I		I

Table 7.6 Sniffing Feedback

Reproduction	Thirst	Hunger	Sex	Existence	Pleasure	Pain	Curiosity
SimOrg:	I	I	D	(I)	D ()	D	I (D)
Predator:	I	I	D	I ()		I ()	D ()
Rock:					I	I	I
Hole:					I	D	I
Water:					I	D	I
Del-Food:					I	D	I
NP-Food:					I	D	I
P-Food:					I	D	I
Bad-Food:					I	D	I

Table 7.7 Reproduction Feedback

In summary, the environment is a square grid that permits the existence of any number of environmental objects at any cell. As a part of a SimOrg's execution sequence, it performs an action on one or more objects at or near its current grid position. The effects of the actions will increase or decrease its drives.

7.2 Plentiful Environment

In this section, we present the results of our model running in the environment shown in figure 7.1. The key for this environment is as follows: *R* is a rock, *H* is a hole, *W* is water, *m* is moldy food, *d* is delicious food, *P* is poisonous food, and *n* is nonpoisonous food. The environment is a 13 by 13 grid, whose boundary, although not shown, is constructed out of rocks.

Figure 7.1 Plentiful Environment

The purpose of the first experiment is to demonstrate that, given a suitable environment (e.g., sufficient food and water) SimOrgs can learn useful knowledge at multiple levels, that evolution can evolve stable, hierarchical structures which encode knowledge, and that the EBR process works.

To establish a SimOrg's performance baseline in this environment - how well it survives - we ran experiments with its EBR process disabled. Without EBR a SimOrg responds with only random actions to its current environmental context and internal motivational state. From running ten experiments, we found that the median change in a population's total age was a 1% decrease, the median age of the population was 22 iterations, the median age of death was 43 iterations, and the median change in a population's age of death was a 2.9% decrease - SimOrgs were dying at an earlier age in later generations.

7.2.1 Species Learning

To show learning at the genotypic level - species learning - we first consider the ongoing development of genes that control the use of planning, analogy and concept schema templates, that control the use of condition dropping, and that define the sizes of long and short-term memory.

Figure 7.2 displays the cumulative number of population members whose plan template, analogy template, concept template, and condition dropping genes are set to true in a moving data collection window that consists of the previous 30 statistics gathering iterations, each of which records data for the system's previous 30 iterations. We have chosen to show the cumulative number of genes set because it provides a better indication of the trends in the population's genes.

Figure 7.2 Template and Condition Dropping Genes That Are Set To True in Test Run #1

In this figure it is clear that the species is moving towards using plan, analogy, and concept templates, and condition dropping. This is what one would expect. For example, SimOrgs that do not use planning templates are forced to rely on randomly produced actions because they can not generate the planning schemata from short-term memory for their EBR process to construct their action solutions. Since SimOrgs that must always rely on random action are very likely to die, it is to the species' advantage to have its members generate reliable schemata from which to plan.

The species also benefits from dropping conditions from its schemata. For example, by dropping conditions, a plan schema may form from two similar, but not identical, short-term memory contexts, because features in the two contexts that don't match are dropped from the resulting plan schema. Condition dropping, if allowed, is also performed when a schema's fitness is determined. In this situation, when matching a schema to the environmental context, a feature will match if it is contained in both, or just one of the contexts. This matching rule is important because general rules produced by dropping conditions will also match those contexts that helped to form them. Since long-term memory is expensive - any increase in its size increases the amount of time spent in EBR processing which in turn increases the hunger drive - the production and subsequent use of a small number of general rules, rather than many specific rules, is to a SimOrg's advantage.

Out of ten test runs, only once did we see that a population was tending to have fewer of its members use planning templates, and in nine test runs we saw increases in the use of condition dropping. These results back up our initial expectations. However, the test runs did not show that the species had a strong preference for using analogy and concept templates. In fact, in only five and six of the test runs did we see that populations had increases in the use of the corresponding templates. These results are summarized in table

7.8, where 'I' means an increasing trend in the gene, 'D' means a decreasing trend, and the values in brackets indicate the percentages of increase or decrease.

Test Run	Plan	Condition Dropping	Analogy	Concept
1	I (22)	I (10)	I (18)	I (5)
2	I (31)	I (71)	I (25)	I (62)
3	I (2)	I (6)	D (9)	I (47)
4	I (91)	I (4)	D (6)	I (31)
5	I (10)	D (3)	D (1)	I (14)
6	I (5)	I (14)	D (65)	D (28)
7	I (19)	I (99)	D (19)	D (13)
8	I (16)	I (23)	I (29)	D (6)
9	I (41)	I (13)	I (6)	I (33)
10	D (10)	I (38)	I (67)	D (8)
Median	I (19)	I (14)	I (6)	I (14)

Table 7.8 Template and Condition Dropping Gene Trends

Interestingly, not one test run showed that the condition dropping, analogy, and concept genes were all decreasing. It would seem then that the species is better with at least one of them increasing. The fact that condition dropping and analogy and concept schemata all play similar roles could explain these increases; all three enable schemata to match better with other schemata. Condition dropping permits missing features when matching, and analogy and concepts schemata add missing features to a schema that enables it to combine with other schemata. Since EBR relies on joining similar schemata together, any process that enables it to produce and use similar schemata is bound to help it.

As previously mentioned, the greater the sizes of long and short-term memories, the greater the amount of time a SimOrg must spend forming schemata, either using EBR or schema templates, which increase its hunger drive. To decrease its hunger drive, a SimOrg has two options: become more successful at food gathering or require less food. To be more successful at food gathering the species could increase its cognitive abilities. This is what it has done; the construction and use of planning schemata and the ability to perform condition dropping are both on the increase in the species. Our results indicate that the species has also taken another approach to decrease its hunger drive. It continually decreases the sizes of both long and short-term memories. The decreases in memory sizes decrease the amount of processing time required by EBR and schemata template formation which decreases a SimOrg's food requirement. Table 7.9 summarizes these results from the ten test runs. In this table, 'I' means an increasing trend in the memory size, 'D' means a decreasing trend, and the values in the brackets indicate the percentages of increase or decrease. In seven of the ten test runs there was a decrease in the size of long-term memory, and in nine cases there was a decrease in the size of short-term memory.

Figure 7.3 displays the changes in the species' memory sizes for test run #2. It is quite clear that the species is decreasing its memory size. To find out by how much, we can calculate the average memory size for each SimOrg by simply taking each y-value in the graph and divide by 330. The y-value is the sum of the memory sizes of 11 SimOrgs (10 plus 1 predator) for the last 30 statistic gathering iterations; therefore, each y-value represents the sum of the memory sizes of 330 individuals. We find the average size of long-term memory decreases from 23 to 19.0, and the average size of short-term memory decreases from 16.2 to 12.0. Although these decreases help a SimOrg to function faster, there is also a cost associated with them: a SimOrg stores less information about its environment and if it does not store enough to reason from, it will be forced to use random

actions which will lead to its early death. However, this does not happen, and, as will be described soon, these changes enable a SimOrg to survive longer.

Test Run	Long-Term	Short-Term
2	D (16)	D (32)
4	D (10)	D (15)
5	D (8)	D (17)
6	D (18)	D (24)
7	D (10)	D (1)
9	D (25)	D (19)
8	I (1)	D (3)
1	I (4)	D (9)
3	I (8)	D (7)
10	D (2)	I (1)
Median	D (10)	D (15)

Table 7.9 Long and Short-Term Memory Trends

Figure 7.3 Sum of the Long and Short-Term Memory Sizes of Test Run #2

So far, our results have indicated that the species is learning appropriate values for its genes. As well as learning contents the species can also learn how to organize genes. To show this we examine the genome's Motivational Drive Hierarchy. The Motivational Drive Hierarchy contains the priorities of a SimOrg's seven drives: existence, hunger, curiosity, pain, pleasure, sex, and thirst. It is organized as a tree where each drive's (gene's) level determines its priority.

We originally expected that the system would evolve a Motivational Drive Hierarchy arranged as in figure 7.4. Since the curiosity drive is satisfied when a SimOrg moves, and movement helps it to explore its environment, we felt it would be better able to satisfy its other drives by always choosing first to move about its environment. After curiosity, we thought a SimOrg should satisfy its hunger and thirst drives. Not satisfying these drives would eventually lead to its death. We thought that a SimOrg would attempt to satisfy the drives of sex, existence, pain, and pleasure last. However, our intuition was wrong. Table 7.10 presents the final levels obtained from each of our ten test runs.

Figure 7.4 Expected Motivational Drive Hierarchy

Test	Existence	Hunger	Curiosity	Pain	Pleasure	Sex	Thirst
7	0	2	2	2	2	1	2
2	0	3	4	1	3	1	2
1	0	2	2	2	2	2	1
8	0	2	2	2	2	2	1
4	1	0	1	1	1	1	1
5	1	0	3	2	2	2	1
10	2	0	3	1	1	2	1
9	2	2	0	3	2	1	2
3	2	2	2	3	3	0	1
6	1	2	2	3	2	0	3
Average	0.9	1.5	2.1	2.0	2.0	1.2	1.5

Table 7.10 Final Drive Levels

By rounding down the averages calculated from table 7.10, we construct the average Motivational Drive Hierarchy shown in figure 7.5.

Figure 7.5 Average Motivational Drive Hierarchy

The highest priority drive in the Motivational Drive Hierarchy is the existence drive. Initially this was somewhat unexpected because we thought that the curiosity drive would be at the top. However, by looking at our environment we found the reason why. In our implementation, if a Predator decided to eat a SimOrg that was beside it, rather than increasing the SimOrg's pain drive, it was killed. Therefore, it was to the species' advantage if its members took immediate action when ever they were beside a Predator. Since a SimOrg that moves into or remains beside a Predator experiences an increase in its existence drive while moving away results in a decrease in the drive, by having that drive at the top of the Motivational Drive Hierarchy it enables the SimOrg to attempt to escape from danger before doing anything else, thus keeping itself and the species alive.

The thirst and hunger drives played similar roles in sustaining a SimOrg's life because if either one reaches a value of 1 the SimOrg is killed. These drives, however, could be satisfied by taking the appropriate action (eating food or drinking water) in many different locations in the environment sometime in the future. A SimOrg has some time to find food or water because these drives are activated before they reach a value of 1 - the value where death occurs. A SimOrg whose existence drive was activated needs to take action immediately because death could occur anytime.

Another surprising result was that the sex drive was at the priority of eating and drinking, and in fact its average level was of greater importance to the species than eating or drinking. Originally it was thought that, because a SimOrg who neglects its sex drive would still survive, while if it failed to eat or drink it would die, its sex drive could be placed at a priority less than eating or drinking. However, a SimOrg who neglects its sex drive does not generate any offspring; therefore, any information stored in its genome is not passed on to the next generation. For a species to survive it is more important that its members reproduce than eat or drink. Whether a SimOrg dies is irrelevant to the species

provided it lives long enough to reproduce. It can always eat or drink later, but when given the opportunity to reproduce it should do so immediately.

The drives of curiosity and pleasure had no immediate influence on whether a SimOrg survives. Although a SimOrg who was curious, and moved about to satisfy its curiosity was bound to discover things, the results indicate that any non-life threatening drive is moved to the bottom of the average hierarchy.

Although our individual test runs converged to many different hierarchies, the average hierarchy provides a good description of the general organization of the species' Motivational Drive Hierarchy. It even pointed out some relationships that were not initially anticipated. The differences in the individual hierarchies demonstrate that more than one hierarchy works well in our environment.

Finally, we examine the population's total age and at what age SimOrgs die in the moving 30 generation window. Since a SimOrg that lives longer is more likely to reproduce and be able to spend more time refining its knowledge, it is to its advantage to increase its life expectancy. In order for SimOrgs to increase their life expectancy they must adapt their existing capabilities and produce new ones that help them to satisfy their drives and survive longer in their environment.

Although the ages of the population's members and the ages at which their deaths actually occurs¹⁶ are not parameters in the genome, the genome's components affect these values, and they are good indicators of how well the species is performing. Table 7.11 contains the results of these values in our 10 test runs. The age and death trends show by how much a population's total age and the average age of death have increased or decreased throughout an experiment. The average age and average death columns display the corresponding values for the last 30 statistic gathering iterations.

¹⁶ There is a parameter in the genome that indicates when death must occur, but a SimOrg usually dies from lack of food or water, or from being eaten, not from reaching the maximum age.

Test Run	Age Trend	Death Trend	Avg Age	Avg Death
1	I (6)	I (10)	25.9	52.3
2	I (4.1)	I (11)	24.1	53.0
3	I (20)	I (33)	29.7	65.5
4	I (9)	I (8)	16.9	33.5
5	I (25)	I (29)	29.0	59.9
7	I (4)	I (10)	25.7	54.7
9	I (19)	I (23)	29.4	60.1
10	I (16)	I (18)	28.9	60.1
6	D (5)	I (2)	21.5	45.6
8	D (3)	I (1)	23.1	49.1
Median	I (9)	I (11)	25.9	54.7

Table 7.11 Population Age and Death Trends

As is clear from this table, both the population's age and the average age at death are increasing; the species' members are living longer, and so the genome must be acquiring values that enable the SimOrgs to survive better in their environment. As an example, consider the population's total age and the age of death shown in figure 7.6 for test run #1. Since the population's total age displayed in the figure represents the sum of ages over the last 30 statistic gather iterations, to get the average age of one SimOrg we divide the corresponding value by 330 (10 SimOrgs and 1 Predator for 30 generations). In this example we find that the average age increases from 24.3 to 25.9 iterations - a 7 percent increase. For the same reason as above, to calculate the average age of death we divide the corresponding value by 30 (the number of iterations). In this example we find that the

population member's average age of death goes from 47.9 to 52.3 generations - a 10 percent increase.

The median age of a population member in its last iteration from our ten test runs is 25.9 iterations, and the median age at death is 54.7 iterations. Compared to the corresponding values produced by the performance baseline experiment - EBR process disabled - there is a 17.7% increase in the average age of a population member, and a 27.2% increase in the average age of death. Although small, these increases do indicate the species is succeeding to increase its life span.

**Figure 7.6 Age of Population and Age of SimOrg's Death
of Test Run # 1**

7.2.2 Individual Learning

To demonstrate learning at the cognitive level - individual learning - we first consider the ongoing strength of the population's knowledge. If the species is improving one would expect an increase in this strength. As a measure of strength, we first compare the sums of

the population's positive and negative schema counters. These counters are used by a SimOrg to compute a schema's fitness and they are a good indicator of whether schemata are being used successfully. Table 7.12 displays the trends of these counters in our 10 test runs. The values in brackets are the percentages of increase (I) or decrease (D).

Test Run	Negative Counter	Positive Counter
1	I (39)	I (01)
3	I (23)	I (45)
4	I (07)	I (06)
5	I (68)	D (04)
9	I (95)	D (20)
10	I (25)	D (16)
2	D (09)	D (42)
6	D (39)	D (59)
7	D (23)	D (32)
8	D (12)	D (04)
Median	I (07)	D (04)

Table 7.12 Positive and Negative Schema Counters

At first glance it would appear that there is a problem with the population's schemata. In 6 out of ten test runs, the negative count of the schemata increases, and in 7 out of ten test runs, the positive count decreases. What is the problem?

We saw in table 7.9 and in figure 7.3 that there were trends to decrease the sizes of long and short-term memories. One effect of this is that long-term memory can not hold as many schemata, thus driving down both the number of good schemata it can hold and the positive count. Decreasing the size of short-term memory results in less information to

build schema using schemata templates. In fact, if short-term memory is too small, a template may not be able to find any matching contexts. This results in a decrease in the production of schemata for use by EBR, which, in addition to the decrease in the number of schemata to work with as a result of decreasing long-term memory, also forces EBR to work with small numbers of new schemata produced from templates.

Even though there was a decrease in the positive count and an increase in the negative count, our median result indicates that the positive count is 2.7 times greater than the negative one. This can also be seen in figure 7.7, which shows the progression of these counters for test run #10.

Restricting memory size increases a SimOrg's efficiency - it no longer requires as much food - but the cost of the restriction is the reliability of knowledge. Even so, SimOrgs manage to develop useful information and use it to their advantage in the environment. If they were not doing this, they would die at the age found in the baseline performance experiment. However, we see how they are in fact living longer.

Figure 7.7 Positive and Negative Schemata Counters of Test Run #10

Next we examine the species' use of evolved versus randomly produced knowledge. If the species' members are producing fitter knowledge, their EBR processes should be able to rely on it more and adapt it to solve their problems. As can be seen in table 7.13, this is exactly what the species is doing. In nine out of ten test cases, evolved knowledge is used more often (median 39.7% more) than random knowledge. And in 7 out of those nine test cases, the use of random knowledge was on the decrease, while the use of evolved knowledge was on the increase. In test cases 1, 4, and 8, the use of evolved knowledge was on the decrease, and I suspect this resulted from the corresponding decreases in memory sizes. The less memory there is, the less schemata it can contain, and, as a result, EBR must often make many changes to existing schemata in order to produce a solution to a problem. As the number of adaptations to a schema increases, the less reliable it becomes.

Test Run	Random	Evolved	Usage
1	I (02)	D (02)	Evolved (30%)
4	I (08)	D (10)	Random (31.5%)
8	I (01)	D (01)	Evolved (11.3%)
2	D (24)	I (19)	Evolved (84%)
3	D (20)	I (17)	Evolved (62.2%)
5	D (10)	I (10)	Evolved (17.4%)
6	D (35)	I (44)	Evolved (77.9%)
7	D (20)	I (21)	Evolved (39.7%)
9	D (07)	I (07)	Evolved (46.3%)
10	D (07)	I (07)	Evolved (8.7%)
Median	D (10)	I (10)	Evolved (39.7%)

Table 7.13 Evolved verses Randomly Produced Schemata

As an example of the growth of the use of evolved knowledge consider figure 7.8. In this figure it is clear that evolved knowledge is used more often than knowledge that is randomly produced, and that the use of evolved knowledge is on the increase.

Finally we examine some of the knowledge produced by older members of the populations in their final iterations. For example, older SimOrgs in our test runs - age above 50 iterations - were able to produce the following schemata:

PlanSchema #1

Conditional context:

(smell(bad), taste(sweet), sight(green)), texture(cold), drive'(hunger))

"NonPoisonous Food"

Action: (eat)

Expected Context:

(smell(bad), taste(sweet), sight(green), texture(cold), drive"(hunger))

"NonPoisonous Food"

PlanSchema #2

Conditional Context:

(texture(cold), smell(bad), taste(sour), sight(blue), drive'(hunger))

"Poisonous Food"

Action: (eat)

Anticipated Context:

(texture(cold), smell(bad), taste(sour), sight(blue), drive"(hunger))

"Poisonous Food"

Figure 7.8 Evolved versus Randomly Produced Schemata in Test Run #2

Plan Schema #3

Conditional Context:

((sight(blue), hearing(loud), texture(cold))) "Water"

((sight(gray), texture(cold))) "Rock"

(drive'(thirst))

Action: (drink)

Anticipated Context:

((sight(blue), hearing(loud), texture(cold))) "Water"

((sight(gray), texture(cold))) "Rock"

(drive"(thirst))

Plan Schema #4

Conditional Context:

(sight(green), taste(sour), texture(warm), hearing(quiet), smell(bad))

"Predator"

(sight(blue), texture(cold))

"Poisonous Food"

(drive'(existence))

Action: (MoveNorth)

Anticipated Context:

(sight(gray), texture(cold), drive"(existence))

"Rock"

Concept Schema #1

Conditional Context:

(sight(gray), texture(cold))

"A Rock"

Anticipated Context:

(smell(bad), taste(sour), sight(green), texture(cold)) "Moldy Food"

Concept Schema #2

Conditional Context:

(smell(bad), taste(sweet))

"NonPoisonous "

Anticipated Context:

((sight(black), texture(none))

"Hole"

Analogy Schema #1

Conditional Context:

(sight(blue), drive(sex))

"Poisonous Food"

Anticipated Context:

(sight(green), taste(sweet), smell(bad), drive{sex)) "Delicious Food"

Analogy Schema #2

Conditional Context:

(texture(cold), sight(gray), drive(hunger)) "Rock"

Anticipated Context:

(smell(good), taste(sour), hearing(loud), texture(warm),
sight(white), drive(hunger)) "SimOrg"

A plan schema asserts that if its action is taken when its source context's conditions are satisfied, then the expected context will be obtained. In each sample plan schema the drive's activity value in the expected context (drive'(*)), although not shown, is less than the value in the conditional context (drive'(*)). Therefore, performing a schema's action also asserts that the drive's activity value will decrease.

What is interesting about plan schemata #1 and #2 is that the feature of food not making any sound has been dropped from the contexts. This is a reasonable feature to drop, since all types of poisonous and nonpoisonous food make no sound, and it is, therefore, not a discriminating feature. Although eating poisonous food increases a SimOrg's pleasure and pain drives, the ingestion of any type of food decreases its hunger drive, and these two schemata are both solutions to a SimOrg's hunger.

Plan schema #3 provides an early description of the notion of how to satisfy thirst. The plan indicates that when a SimOrg is by rocks and water, if it drinks both, this will decrease its thirst drive. The problem for it to figure out is which of the two will lead to a larger decrease in its thirst drive.

Plan schema #4 instructs a SimOrg to move north when it is beside poisonous food or predators. This is helpful to a SimOrg in two ways: it moves away from predators before being eaten, and it also moves away from poisonous food which is harmful to eat.

A concept schema implies that its component features are causally related to one another.

What is interesting about the Concept Schema #1 is that it makes the connection that moldy food often occurs near rocks. This is not obvious from the environment figure 7.1 because the boundary of rocks around the environment is not shown, but the rocks are there, and the moldy food is mainly at the boundary where the rocks are. In Concept Schema #2 the connection is made that food can be found near holes. From the environment one can see that by the holes both types can be found. The concept has dropped the unmatched features between the types of food to produce the concept of where food can be found.

An analogy schema asserts that the two contexts can be interchanged with one another.

Analogy schema #1 forecasts that having sex with poisonous food is the same as having sex with delicious food. Of course having sex with food does not decrease an entity's sex drive, but it does indicate that poisonous food and delicious food may be considered the same item. When a SimOrg is hungry and is near poisonous food, by combining this schema with another one that says it is all right to eat delicious food, a SimOrg might eat the poisonous food. Although this increases its pain, it will also decrease its hunger drive, which in the short-term may be just what it needs to survive.

Analogy schema #2 forecasts that eating a rock will have the same consequences as eating another SimOrg. They are similar since eating either one increases a SimOrg's hunger and thirst drives.

There are many more schemata that one could select, but the ones shown are good examples of interesting information produced by our model. What is missing from these examples are complex schemata. We found that it took a considerably greater number of

generations for complex schemata to form: over 500 generations. Since SimOrgs never lived beyond 70 generations, they never formed complex schemata. In fact, it took almost 40 generations for them to begin to build any reliable schemata. As the results of the STGA showed, one needs to build a stable foundation of knowledge before it can be combined to form higher-level knowledge, so this result is to be expected.

In the later environments we perform experiments where SimOrgs are given infinite life which allows them the time to develop complex schemata. First, the SimOrg's parameters and cognitive abilities set correctly for the environment. Then they operate as before but without the possibility of dying. Although their drives still change values, no SimOrg is killed if, for example, its hunger drive reaches its maximum value. This enables SimOrgs to continually build up their knowledge about the world and form complex schemata.

In this section, we have demonstrated that in a simple environment the model performs as it is supposed to. Applying the process of evolution at three distinct levels to objects from motivational hierarchies and SimOrgs to schemata, the model acquires information about the world that is appropriate for each level.

7.2.3 SimOrgs With Sensory Deficiencies

In our next experiment, SimOrgs are forced to use the Average Motivational Drive Hierarchy found in figure 7.5, and their senses of hearing and touch permanently disabled in the plentiful environment. The purpose of this experiment is to show the effects the lack of two senses have on a SimOrg and show that it can still adapt and thrive with the handicap.

To examine learning at the genotypic level - species learning - we consider the ongoing development of the plan template, analogy template, concept template, and condition dropping genes. Table 7.14 displays the cumulative number of population members whose plan template, analogy template, concept template, and condition dropping genes are set to

true in a moving data collection window that consists of the previous 30 statistics gathering iterations, each of which records data for the system's previous 30 iterations. In the table an 'I' means an increasing trend in the gene, 'D' means a decreasing trend, and the values in brackets indicate the percentages of increase or decrease.

Test Run	Plan	Condition Dropping	Analogy	Concept
2	D (4)	I (138)	I (7)	I (21)
6	D (11)	D (6)	I (11)	D (10)
9	D (29)	I (10)	I (12)	I (33)
3	I (13)	D (31)	D (23)	I (0)
8	I (12)	D (15)	D (-2)	I (13)
1	I (1)	I (8)	D (1)	I (49)
10	I (2)	I (21)	D (27)	I (42)
4	I (38)	I (28)	I (18)	I (51)
5	I (17)	I (31)	I (36)	I (22)
7	I (48)	I (11)	I (12)	I (23)
Median	I (12)	I (11)	I (11)	I (23)

Table 7.14 Template and Condition Dropping Gene Trends

Similar to the original test runs in the plentiful environment, we see that the population members tend to use planning and condition dropping templates. These results back up our initial expectations of section 7.2.1. However, the new test runs also show that the species has a stronger preference for using analogy and concept templates than in the original test runs. Although the condition dropping, analogy and concept schemata all play similar roles, the absence of the two senses has increased the selective advantage for a SimOrg that has all of these genes turned on.

In the original set of test runs, the species decreased its long and short-term memory sizes, to conserve energy when using EBR, as shown in table 7.9. In the absence of the senses of hearing and touch, the test runs shown in table 7.15 indicate that the species has allowed long-term memory to become a bit larger, and it has hardly decreased the size of short-term memory. Lacking the information provided by these two senses, the species now requires more information to be stored and processed in order to stay alive, and it is not allowing evolution to decrease the memory sizes.

Test Run	Long-Term	Short-Term
2	D (2)	D (1)
4	D (4)	I (3)
6	D (1)	D (3)
7	D (4)	I (1)
1	I (3)	D (2)
3	I (1)	D (6)
5	I (8)	D (2)
8	I (4)	D (2)
9	I (4)	I (2)
10	I (3)	I (1)
Median	I (3)	D (2)

Table 7.15 Long and Short-Term Memory Trends

Similar to the original test runs, the test run results shown in table 7.16 show both the population's age and the average age at death increasing; the species' members are living longer, and so the genome must be acquiring values that enable SimOrgs to survive better in their environment.

The median age of a population member in its last iteration from our ten test runs is 21.4 iterations, and the median age at death is 40.5 iterations. Compared to the corresponding values produced by the performance baseline experiment - EBR process disabled - there is a 2.7% decrease in the average age of a population member, and a 5.8% decrease in the average age of death.

Why are there decreases in the median age of a population member and the median age at death? The answer can be found in what the SimOrgs eat. Without a SimOrg's senses of touch or hearing, moldy food and nonpoisonous food differ by only one feature. Moldy food is sour; nonpoisonous food is sweet. This difference leads SimOrgs to often eat moldy food which causes them pain and death. Those SimOrgs that manage to produce schemata that discriminate between the two types of food survive longer. However, because SimOrgs can not communicate with one another, the useful schemata are not passed to other members of the species. A SimOrg that uses random actions will not necessarily eat food when it is hungry. A SimOrg that has previously used a schema to eat good food will also use it for bad food, if no other better schema exists. This result is schemata are used in inappropriate situations causing a SimOrg to experience more pain and earlier death than it would experience using random actions for feeding.

This result can also be seen in the schemata usage counters contained in table 7.17. Even though long and short-term memory sizes changed very little, there was an 11% median increase in the negative schemata counters and a 4% median decrease in the positive schemata counters. However, our median results indicate that the positive count is still 4.5 times greater than the negative one.

Finally we examine the species' use of evolved versus randomly produced knowledge. Similar to the results of the original test runs, although not as large, the median results shown in table 7.18 indicate that the use of random knowledge is on the decrease, while the use of evolved knowledge is on the increase. However, these results also indicate that the

species is relying on more randomly produced knowledge than in the original test runs. As demonstrated by the previous results, this is to be expected. If a SimOrg is having trouble

Test Run	Age Trend	Death Trend	Avg Age	Avg Death
1	I (8)	I (13)	21.4	40.2
2	I (34)	I (6)	21.2	40.4
3	I (2)	D (3)	20.1	36.9
4	I (19)	I (18)	23.2	42.1
5	I (15)	I (18)	25.5	49.3
6	I (1)	I (0)	20.0	37.7
7	I (7)	I (9)	23.0	44.0
8	I (6)	I (2)	21.9	40.5
9	I (1)	I (5)	20.2	39.6
10	I (3)	I (1)	19.9	37.3
Median	I (7)	I (6)	21.4	40.5

Table 7.16 Population Age and Death Trends

discriminating between good and bad food because the schemata that work on one type of food fail on another, it is better off using random actions rather than always using the one that fails.

In this section, we have shown that SimOrgs lacking the senses of touch and hearing perform well in the plentiful environment, but not as well as those with all five senses. The main problem for a SimOrg is that it perceives good and bad food as having closely similar features, and bad food will cause it pain and death when eaten. The species' solution is to not always rely on evolved knowledge and, instead, rely more on random actions. Since few SimOrgs live long enough to develop the correct schemata to discriminate between the

<u>Test Run</u>	<u>Negative Counter</u>	<u>Positive Counter</u>
6	I (01)	I (02)
7	I (47)	I (07)
8	I (03)	I (01)
1	I (11)	D (04)
3	I (11)	D (07)
4	I (34)	D (04)
5	I (46)	D (04)
2	D (06)	D (14)
9	D (12)	I (01)
10	D (07)	D (01)
Median	I (11)	D (04)

Table 7.17 Positive and Negative Schema Counters

<u>Test Run</u>	<u>Random</u>	<u>Evolved</u>	<u>Usage</u>
1	D (07)	I (08)	Random (01%)
2	D (12)	I (86)	Evolved (81%)
3	D (06)	I (14)	Random (05%)
4	D (08)	I (09)	Evolved (01%)
5	D (01)	I (01)	Evolved (08%)
9	D (24)	I (29)	Evolved (43%)
10	D (05)	I (05)	Evolved (06%)
6	I (03)	D (03)	Random (08%)
7	I (03)	D (03)	Evolved (2%)
8	I (2)	D (02)	Random (03%)
Median	D (05)	I (08)	Evolved (02%)

Table 7.18 Evolved verses Randomly Produced Schemata

two types of food, the best solution would be for them to pass that knowledge to their offspring. However, since SimOrgs can not communicate this is impossible, but this does show that the selection pressure is present for communication to evolve.

7.2.4 Alternate Environmental Objects

In our next experiment, each SimOrg is forced to use the Average Motivational Drive Hierarchy in the plentiful environment with one modification: all instances of Delicious Food are removed from it. This makes Nonpoisonous Food the one and only suitable food source for a SimOrg in the environment. This change is only temporary. After completing half of a test run's generations (375 deaths), all instances of Nonpoisonous Food are replaced with Delicious Food, making Delicious food the only suitable food source. The purpose of this experiment is to see if and how a SimOrg adapts to a changing environment and a decrease in the food supply.

To examine learning at the genotypic level, we consider the ongoing development of the plan template, analogy template, concept template, and condition dropping genes. Table 7.19 displays the cumulative number of population members whose plan template, analogy template, concept template, and condition dropping genes are set to true.

Similar to the initial plentiful environment median results, the setting of condition dropping, analogy, and concept genes to true is on the increase in the species, and not one test run showed these genes simultaneously decreasing. It would seem then that the species is better with at least one of them increasing. The results also show that when the species attempts to decrease the construction of plans from short-term memory using schemata templates, it most often increases the use of condition dropping, analogy and concepts, and visa versa. This is to be expected. If the species is not building plans using schemata templates and relying on randomly generated plans for the source building blocks of the

EBR process, it must rely more on condition dropping and analogies during the schemata matching process to make better matches and use of the source schemata.

Different from the results of the last two sections, the building of plans using schemata templates is on the decrease. Since the time spent searching for plans in short-term memory increases a SimOrg hunger drive, not looking for plans can be seen as another attempt by the species to decrease its food requirement. If by dropping conditions and using analogies the species can survive better with randomly generated plans, the genotypic level has acquired a reasonable setting for the corresponding genes.

Test Run	Plan	ConditionDropping	Analogy	Concept
1	D (05)	I (15)	I (16)	D (13)
2	D (18)	I (30)	I (10)	I (6)
4	D (11)	I(10)	I (02)	I (15)
6	D (04)	I (13)	I (23)	I (15)
10	D (20)	I (04)	I (10)	I (7)
7	D (22)	D (08)	I (02)	I (33)
9	I (30)	I (30)	I (30)	I (10)
8	I (04)	D (15)	D (14)	I (09)
5	I (13)	I (1)	D (04)	I (10)
3	I (2)	I (14)	D (27)	D(2)
Median	D (04)	I (13)	I (10)	I (10)

Table 7.19 Template and Condition Dropping Gene Trends

The changes to the sizes of long and short-term memories also differ from the previous results. In this experiment there is a modest increase of 1% in the size of long-term memory and a decrease of 1% in the size of short-term memory, shown in table 7.20. We find the

average size of long-term memory is 22 and the average size of short-term memory is 13.0. In the initial results, the decreases were 10 and 15 percent respectively, which were seen as attempts at decreasing a SimOrg's food requirement. In this experiment, since SimOrgs are relying less and less on plans created with planning templates in short-term memory, short-term memory's size is not a factor. Memory storage costs a SimOrg nothing because it is the processing time spent scanning it with templates that increases its hunger drive. The small median change in the size of short-term memory reflects this fact: a change in size has no selective advantage to the SimOrg.

Test Run	Long-Term	Short-Term
3	D (01)	D (01)
5	D (01)	I (01)
7	D (01)	I (01)
1	I (01)	D (01)
4	I (02)	D (01)
6	I (02)	D (15)
8	I (02)	D (01)
9	I (01)	D (01)
2	I (02)	I (02)
10	I (01)	I (01)
Median	I (01)	D (01)

Table 7.20 Long and Short-Term Memory Trends

I believe that the reason for the observed increase in the size of long-term memory is a result of it initially containing randomly generated plans rather than ones formed from the application of schemata templates to short-term memory. Since randomly produced

schemata are less reliable, it is to the SimOrgs advantage to have more of them for the EBR process to work with.

Next we examine the population's total age and at what age SimOrgs die. As previously argued it is to the species advantage to increase its life expectancy, and in the initial results SimOrgs did just that. In this experiment, the species also manages to do the same. However, the average age of the population and the average age of a SimOrg at death are less than those found in the initial experiment. It was noted that the majority of deaths were caused from a lack of food. These results are to be expected because a SimOrg's task of finding food in this experiment is much more difficult because of the 50% decrease in the food supply. As shown in table 7.21, even with the decrease in food supply both the species' age and death trends still show increases of 4 and 5 percent respectively, and, although small, these increases do indicate the species is acquiring information that makes it more successful in the environment.

Test Run	Age Trend	Death Trend	Avg Age	Avg Death
1	I (01)	I (01)	19.1	35.4
2	I (01)	I (05)	19.8	38.0
3	I (07)	I (03)	21.1	39.0
4	I (06)	I (09)	22.1	42.3
6	I (19)	I (34)	19.4	54.6
7	I (01)	I (01)	20.8	38.0
8	I (05)	I (06)	19.3	34.4
9	I (04)	I (07)	21.5	40.0
10	I (03)	I (02)	19.4	35.2
5	I (01)	D (01)	20.6	38.6
Median	I (04)	I (05)	20.6	39.0

Table 7.21 Population Age and Death Trends

At the cognitive level we first examine the ongoing strength of the population's knowledge, defined by its member's positive and negative schema counters trends, shown in Table 7.22. In this experiment long-term memory experienced only a 1% increase in size, which enables every SimOrg to store only one more schema. This small increase in storage results in the small increases the SimOrgs experience in their positive and negative schemata counters. Although only small increases, our median result indicates that the positive count is 4.6 times greater than the negative one. This indicates that SimOrgs are making better use of their schemata than in the initial experiment, saving only those schemata that help them to survive in their food deprived environment.

Test Run	Negative Counter	Positive Counter
1	D (03)	I (08)
9	D (02)	D (08)
2	I (05)	D (06)
4	I (20)	D (08)
8	I (08)	D (07)
3	I (14)	I (08)
5	I (05)	I (01)
6	I (65)	I (11)
7	I (13)	I (02)
10	I (05)	I (02)
Median	I (08)	I (02)

Table 7.22 Positive and Negative Schema Counters

Next we examine the species' use of evolved versus randomly produced knowledge. As can be seen in table 7.23, in eight out of ten test cases, SimOrgs use evolved knowledge more often (median 6.8% more) than randomly produced knowledge to solve their problems, and the trend in these experiments is to increase its use further. However, the usage of evolved knowledge in the experiment is not as large as in the initial experiment; the species is relying on more randomly produced knowledge than in the original test runs. This result is due to the fact that SimOrgs don't use planning templates as often as those in the initial experiment.

Test Run	Random	Evolved	Usage
2	I (03)	D (03)	Evolved (05%)
4	I (03)	D (03)	Evolved (2.6%)
8	I (10)	D (03)	Evolved (22%)
9	I (08)	D (07)	Evolved (3.1%)
1	D (07)	I (06)	Evolved (29.4%)
3	D(05)	I (05)	Evolved (7.5%)
5	D (02)	I (02)	Evolved (6.8%)
6	D (03)	I (03)	Random (04%)
7	D (02)	I (02)	Random (08%)
10	D (04)	I (03)	Evolved (29.4%)
Median	D (02)	I (02)	Evolved (6.8%)

Table 7.23 Evolved verses Randomly Produced Schemata

In this section, we have shown that SimOrgs perform well in an environment that changes over time and does not have as much food as the plentiful environment, but not as well as those in the initial plentiful environment. The main problem for a SimOrg is that

food is harder to find in this environment. The species' solution is to not always rely on evolved knowledge and, instead, rely more on random actions. Since few SimOrgs live long enough to develop the correct schemata to locate and eat the food, the best solution is for them to pass that knowledge to their offspring. However, since SimOrgs can not communicate this is impossible, but the selection pressure is present for communication to evolve.

7.3 Spacious Environment

In this section, we present the results of our model running in the environment shown in figure 7.9. This environment is a stretched version of the plentiful environment shown in figure 7.1 and they share the same key. The numbers in the figure (0 - 9) represent SimOrgs. The major differences between the two environments are that this one is a 27 by 23 grid, a 313% increase in area over the first environment, and that this one contains no predator.

The purpose of this experiment is to show what effects a larger environment has on the model's ability to learn useful knowledge at multiple, cooperative levels, its ability to evolve stable, hierarchical structures which encode knowledge, and its EBR process.

To establish a SimOrg's performance baseline in this environment - how well it survives - we ran experiments with its EBR process disabled. From running ten experiments, we found that the median change in a population's total age was a 1.2% increase, the median age of the population was 19.9 iterations, the median age of death was 37.5 iterations, and the median change in a population's age of death was a 1.2% decrease - SimOrgs were dying at an earlier age in later generations.

For the most part, our results from this environment are the same as those for the plentiful environment. The use of planning schema templates was on the increase in seven out of ten experiments (median increase 13.3%). There was no clear indication that analogy

3 of/de 3

MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS
STANDARD REFERENCE MATERIAL 1010a
(ANSI and ISO TEST CHART No. 2)

or concept schemata or condition dropping were of any use to a SimOrg, given that it was using planning schema templates. For example, only half of the experiments showed an increase in the use of both types of schemata and condition dropping. Short-term memory was showing a marginal decreasing trend (median change 0.7%) in the experiments. And the number of experiments with increases in long-term memory size equaled the number with decreases. The schemata's positive counters decreased in half of the experiments, and the negative counters were showing an increasing trend in eight out of ten experiments (median change 11.6%). The positive counter is still greater than the negative one, and evolved knowledge is used more often than randomly produced knowledge. The population's total age was showing an increasing trend in 7 out of ten experiments (median change 2.8%), as was the age at death in 8 out of ten experiments (median change 4.5%).

Figure 7.9 Spacious Environment

The median age of a population member in its last iteration from our ten test runs is 22.3 iterations, and the median age at death is 40.0 iterations. Compared to the corresponding values produced by the performance baseline experiment - EBR process disabled - there is a 12.1% increase in the average age of a population member, and a 6.7% increase in the average age of death. Although not as large as those increases in the plentiful environment, these increases still indicate that the species is succeeding to increase its life span but it is finding this environment more difficult to survive in than the last one.

The main differences between our results of this experiment and those of the plentiful environment are the changes in the average Motivational Drive Hierarchy. Since in this environment there is no predator, nothing could eat a SimOrg and its existence was assured provided it could meet its food and water requirements. In this environment, we found that the existence drive had dropped from its top priority position at level 0 to level 2. The drives at the first level consisted of hunger, thirst, sex, and curiosity, with each of the first three drives taking turns at being the highest priority drive at level 0. Since movement decreases the curiosity drive, the elevation of it in the Motivational Drive Hierarchy demonstrates that SimOrgs are considering it more important to move about in their environment. Since the environment is larger, and to find food and water a SimOrg should move about more, the increase in the level of this drive is to be expected.

To see if SimOrgs would continue to improve their knowledge if they were permitted to live longer, we initialized a population of SimOrgs with their planning, analogy, concept, and condition dropping genes set, a Motivational Drive Hierarchy consisting of hunger at level 0, sex, thirst, and curiosity at level 1, and the other drives at level 2, and all other parameters randomly initialized within their legal limits. This population was given infinite life, meaning that maximum levels of hunger and thirst could not kill its members. The positive and negative schema counters of this population are shown in figure 7.10. From

this figure, we see that the SimOrgs do continue to improve their knowledge; the positive counters are growing at a faster rate than the negative counters, meaning that the knowledge is used successfully more often than not.

Figure 7.10 Positive and Negative Schema Counters

This increase in life span also allowed complex schemata to form. In the last section we saw that with, a short life span, SimOrgs were just beginning to establish a rudimentary level of knowledge when they died. With unlimited life, once this level is established, a SimOrg starts to combine reliable schemata, forming complex ones. The following are examples of the complex plan schemata that were formed and used successfully by SimOrgs.

Complex Schema #1

PlanSchema

Conditional Context:

(sig: t(blue), texture(cold), drive'(thirst)) "Water"

Actions: (drink, drink)

Anticipated Context:

(sight(blue), texture(cold), drive"(thirst)) "Water"

Complex Schema #2

PlanSchema

Conditional Context: (drive'(curiosity))

Action: (MoveNorth, MoveEast, MoveNorth)

Anticipated Context: (drive""(curiosity))

Complex Schema #3

PlanSchema

Conditional Context:

(hear(loud), drive'(thirst)) "Water".

Action: (MoveWest, Drink)

Anticipated Context:

(hear(loud), drive"(thirst)) "Water".

Complex schema #1 instructs a SimOrg to drink twice when it finds water. This is an important schema for two reasons: it is very good at decreasing a SimOrg's thirst, and it decreases the amount of reasoning it must do. The SimOrg could use the single drinking plan schema twice; however, this would cause it to use EBR twice, and there is a hunger cost to pay for each invocation. By using the complex schema, the EBR process is invoked only once to return the complex schema.

Complex schema #2 combines three movement schemata for the purpose of decreasing the curiosity drive. The interesting feature of this schema is that the conditional and

anticipated contexts contain only the active drive. The SimOrg will move north, east, and north when it is curious, no matter what objects are around it. This schema is beneficial to the SimOrg because it does not have to be in any particular area of the environment, or close to any object, for it to use the schema. Those extra conditional features have been dropped. Similar complex plan schemata were also produced with other movement actions.

Complex schema #3 combines two different types of planning schemata together. The schema states that when a SimOrg is thirsty and it hears a loud noise it should move west and then drink. Since only water or another SimOrg makes a loud noise, it will either drink water, drink a SimOrg, or find that the noise (SimOrgs) has gone away. Since SimOrgs are often found near food or water (as in Figure 7.9), even if the SimOrg only hears another one it may still prove useful to go to where it is and drink.

In this section, we have demonstrated that even in a larger environment the model performs as it is supposed to. First, when the predator was taken from the environment, the model made the appropriate adjustment in the Motivational Drive Hierarchy, moving the existence drive to a lower priority. Second, with the increase in size of the environment, the curiosity drive moved up a level. Finally, given enough time, SimOrgs continue to evolve more reliable information, which permits complex schemata to form.

7.4 Hazardous Environment

In this section, we present the results of our model running in the environment shown in figure 7.11. This environment is the same size as the plentiful environment shown in figure 7.1 and shares the same key. The environment has separated the food and water by a wall of rocks with only three openings. Since SimOrgs require both food and water, and since they can not walk through rock walls, they must find their way back and forth between the openings to feed and drink or suffer from hunger or thirst. To make it even

harder for a SimOrg to eat, holes have been placed in front of the food. Like the spacious environment, there is no predator in this environment.

The purpose of this experiment, repeated six times, is to show what effects a difficult environment will have on the model's abilities.

Figure 7.11 Hazardous Environment

There were some similarities in the results of our experiments using this environment and those using the other environments. There was no clear indication that condition dropping or analogy and concept schema templates were of any use to a SimOrg. Condition dropping increased in four out of six experiments (median change 15%). The use of analogy schema templates decreased in four out of six experiments (median change 10.7%), and the use of context schema templates increased in four out of six experiments

(median change 2.8%). Both the sizes of long and short-term memories had very small decreasing trends (a 0.04% median change in long-term memory, and a 0.2% in short-term memory). The schemata's positive and negative counters were both showing decreasing trends in five and six out of six experiments respectively, with median changes of 10.4% and 30.3%. The positive counter was still greater than the negative one in all experiments.

There were four significant differences in the model's behavior in this environment. First, the use of planning schema templates decreased in four out of six experiments (median change 9.3%). Second, the average Motivational Drive Hierarchy placed every drive at the same level. Third, the population's total age and age of death had decreasing trends in five out of six experiments, with median changes of 4.4% and 2.1% respectively. Fourth, the use of evolved knowledge was declining, while the use of randomly produced knowledge was increasing, and in fact was used three times more often than evolved knowledge. This result is shown in figure 7.12.

From these results, it is clear that SimOrgs are not surviving as well in this environment as they did in the last two. They can not determine which templates to use. They can not get their Motivational Hierarchy organized correctly for the environment. They are not surviving very long, and they are relying on mostly randomly produced actions.

It was apparent that in a SimOrg's limited life time it could not produce reliable knowledge to operate in this environment. However, we suspected that, given enough time, a SimOrg could learn the appropriate information. We gave a population infinite life and initialized it the same way as the infinite life population of the previous section. We wanted to know if a SimOrg could reverse the trend and rely more on evolved knowledge.

The results in figure 7.13 show that SimOrgs could reverse the trend, given long enough life spans. In this figure we see that the use of evolved knowledge is increasing while randomly produced knowledge is on the decrease. However, in not one experiment did a SimOrg come to use evolved knowledge more often than randomly produced

knowledge. The problem was that its long-term memory was not large enough to store all of the information it required to survive in the environment. By the time it learned about drinking, it needed to eat; and in the process of learning how to do that, it forgot all that it had learned about how to drink. Then when learning about drinking, again, it forgot what it had learned about eating, and so on.

Figure 7.12 Evolved versus Random Actions in Hazardous Environment

Figure 7.13 Evolved versus Random Actions in Hazardous Environment with Infinite Life SimOrgs

In this section, we have shown that in complex environments SimOrgs require increases in life span and memory to evolve the appropriate knowledge. This is to be expected. Complex plans rely on stable, reliable plans at the level below them, and at each new level there are more plans. As the results of the STGA have shown, it takes longer and longer to evolve each new level of complexity.

7.5 Chapter Summary

In this chapter, we present the results of our model running in three different environments. The plentiful environment is small and contains all of the possible environmental objects, including ten entities and one predator. The spacious environment contains no predator and is a stretched version of the plentiful one. The hazardous environment has no predator, is the same size as the plentiful environment and has food and water separated by a wall of rocks with only three openings. The important feature about all of these environments is that a SimOrg's action can not be determined immediately from its sensor information.

Our results have shown that SimOrgs, using evolution at the genotypic level, manage to determine good settings for their genes, such as those for controlling the use of planning, analogy and concept schema templates and condition dropping, and those that define parameters such as the sizes of long and short-term memory and life span. In addition to learning the gene settings, our results show that the species can also learn how to organize genes. In the first two environments, it was clear that the species is learning the correct Motivational Drive Hierarchy.

Our results show that in the first two environments SimOrgs are surviving longer because both the population's total age and the age at death are increasing, and they are performing better than just using random actions. This is partly due to evolution of the

genome, but it was also due to the SimOrg making proper use of the genome's parameters in the environments. It was, however, more difficult for a SimOrg to survive in the spacious environment.

We found that, in the first two environments, the species relies on evolved knowledge rather than on randomly produced knowledge. We saw that the strength of the population's knowledge was continually increasing, although in the hazardous environment it was clear that SimOrgs required more memory and a longer life span to acquire the appropriate knowledge. We also found that a SimOrg is able to establish what objects are, e.g., water, what they are used for, e.g., drinking, and how schemata can be combined to form complex ones.

It is clear that since SimOrgs can not live for ever, the ability to communicate reliable knowledge from one generation to the next would contribute to the species' success in any environment. The main problem for a SimOrg in the hazardous environment is that it always begins with no knowledge about the environment. For example, it does not know what a rock is, what the rock does to it when it runs into one, or what to eat or drink. Only when a SimOrg is able to live forever could this knowledge begin to be developed.

Chapter 8

Summary and Conclusion

In this thesis we develop a biologically motivated approach to cognitive modeling and machine learning that is built on the foundational work done in EE. In chapter 2, we describe three processes (morphogenesis, natural selection and reproduction) and four conditions (variation in the evolving structures, inheritance of the structures from one generation to the next, a differential in the reproductive success of the structures and constancy in the environment in which the structures are operating) that must be present and met in order for evolution to occur. Although necessary, these processes and conditions are not sufficient for evolution to show progress. For example, in chapter 4 the RGA failed to locate the correct answer even when the processes and conditions were met. To produce a fully functioning evolutionary system, much more research is required. Questions like the following must be answered:

- What structures should encode the knowledge?
- How should these structures be manipulated so they resemble their parent structures, but provide enough variation for evolution to work?
- How is a structure's fitness determined?
- What parameter values should the system use?

- How are evolution's three fundamental processes combined to produce a new generation?

Without answers to these questions, an evolutionary system's structures will not progress, they will not become fitter, and the system will not work. Our evolutionary approach to cognitive modeling answers these questions, and in the process demonstrates evolution to be a fundamental learning process. Its contribution is a fully described and implemented evolutionary cognitive model and machine learning method that uses both symbolic and subsymbolic memory structures.

In the following sections, we elaborate on CompEE's achievements and how it overcomes the disadvantages of the systems mentioned in chapter 1. The final section describes the open issues in CompEE and discusses its future research direction.

8.1 CompEE's Achievements

CompEE's first achievement is the demonstration that learning - evolutionary changes to an individual's internal organization that improves its performance - occurs at multiple levels, each of which uses a different set of structures to encode its knowledge, but all benefiting from the collective knowledge.

To do this, a SimOrg is constructed from three distinct levels: the genotypic level, the phenotypic level, and the cognitive level. Its genotypic level contains a single genome composed of a chromosome hierarchy and a motivational drive hierarchy; its phenotypic level contains four high level systems: the Motivational System, the Stimulus System, the Memory System, and the Problem-Solving System; and its cognitive level, using EBR, produces schemata. A SimOrg is placed in an environment in which its every action either increases or decreases its motivational drives. Its problem is to satisfy its drives and remain alive to reproduce. In order to survive longer, a SimOrg learns the appropriate settings of

its parameters, contained in its genes, for operating in the environment, it learns the correct homeostatic relationships between its systems and it learns reliable schemata for producing correct actions in order to solve its problems. Each of its levels acquires information that is useful for itself and the others to progress. For example, having the planning gene set enables SimOrgs to construct and use planning schemata, which enables them to live longer, which enables new genes such as condition dropping to be set and tried. The genotypic level provides new features for the cognitive level, which, when successful, enables the genotypic level to try new features.

Our results indicate that SimOrgs are indeed improving using our multi-level model: genes adjust themselves to provide new or better features for the SimOrgs; they survive longer with the improved features; and their knowledge improves the longer they live which enables them to reproduce more, passing successful features to the next generation.

CompEE's second achievement is the demonstration that evolution can be used as a universal process for knowledge gain. Rather than using the sequential binary structures of traditional GAs, our approach evolves the contents and organizations of different dynamic, hierarchically organized, symbolic structures. By providing the appropriate genetic operators and fitness criterion, we acquire knowledge in the form of trees, SimOrgs and schemata. The diverse forms and nature of the knowledge acquired: control parameters, drive settings, SimOrgs, plans, analogies, and concepts, shows that the process of evolution can be applied successfully to many different types of knowledge for the purpose of learning.

CompEE's third achievement is its use of hierarchies to evolve complex structures, including knowledge. In comparing the results of the RGA with those of the TGA, it is obvious that the TGA's use of hierarchical structures is superior. In fact, we found that, when possible, evolving a hierarchical structure's components separately (e.g. the STGA and schemata) will produce even better results. Our results show that SimOrgs can evolve

the correct motivational drive hierarchy for determining drive priorities, and can evolve reliable schemata of different types and sizes for solving the problems created by the drives. Both these types of structures are hierarchically organized. As the RGA demonstrates, a sequential structure will not often work in an environment where the specifics of the fitness criterion are unknown, and where a structure's components that determine fitness may have several epistatic relationships (e.g., the TGA and the motivational drive hierarchy). However, the TGA and the STGA work in such an environment. A hierarchy is not only a structure that can be used to evolve correct values, its organization can also be part of the evolving information.

CompEE's fourth achievement is the production of reliable, symbolic knowledge using evolution, forming a new model of reasoning called Evolution-Based Reasoning. Since a SimOrg's long-term memory is a fixed size, and not large enough to contain exact plans for every one of its possible problems given different environmental contexts, the only way its knowledge can show improvement is to use it successfully in multiple situations. For a SimOrg to reuse past solutions, it must adapt them to meet its current problems. This presupposes that its past solutions are valid. It is unlikely that adaptations of invalid solutions will produce valid ones. EBR adapts a SimOrg's existing knowledge to produce solutions to its new problems. Our results show that a SimOrg, using EBR, does acquire and improve its knowledge during its lifetime, and, if given unlimited life, continues to do so.

CompEE's final achievement is the identification of an approach to managing the 'preprogramming' problem of ML - the hand-coding of knowledge and control parameters. Resigning ourselves to the fact that it is unlikely one can ever determine the proper settings to all of a system's parameters, we define them in the genome and have the system discover their values. Our approach considers the task of setting the system's parameters as just another level in the learning system, and applies the same techniques (evolution) for

learning them as is done for learning the structures in the other levels of the system. Knowing that certain parameters may have epistatic relationships, we use hierarchies of parameters to permit these relationships to form. Although the system can not create its own parameters, our results show that it certainly can adjust the ones we give it, and adjust them in such a way that the new settings continually improve the system's performance in succeeding generations.

To attack the problems of symbol grounding and brittleness in symbolic AI, CompEE works in a sensory environment where SimOrgs experience varying degrees of need satisfaction. By working with symbolic representations which are sensory projections of environmental objects, connecting symbols used in internal reasoning to external physical stimuli is no longer a problem. Although we are using only simulated sensors, if placed in a robot with real sensors, our model would still operate the same, making use of the symbolic information produced by its sensors.

Rather than working only with exact algorithmic processes like deductive reasoning, CompEE uses abstract, natural abilities like adaptation. In fact, where many symbolic AI systems consider only one ability, CompEE works with many. For example, it combines the use of condition dropping, the use of analogies and concepts, and the use of EBR - CBR with adaptation. Since it is never clear which combination of abilities are best suited for a given environment, CompEE locates and uses the preferred ones. Therefore, CompEE, in addition to solving problems for a given environment, also tells us something about it.

To attack the static adaptation problem of CBR, resulting from CBR's reliance on hand-coded rules, EBR relies on only four genetic operators: mutate, crossover, join, and split for the adaptation of schemata. Like CBR, EBR still uses past knowledge for its solutions to new problems. However, by randomly applying the four genetic operators, EBR has the facility to look for new and better solutions because its operators do not presuppose when

they should be used or on what types of knowledge. By adapting only the fittest schemata - those that match the current conditions - the exploration is controlled by exploiting only those schemata that have a reasonable chance of solving the new problem. The controlled randomness of EBR, the hallmark of GAs, makes it a much more flexible, opportunistic learning method than CBR.

The subsymbolic techniques of neural networks and GAs have proven that they have many advantages over symbolic techniques. CompEE uses subsymbolic techniques, but to be more transparent, CompEE uses them on the structures normally associated with symbolic techniques. This enables CompEE to be used in domains that require high-level knowledge processing. CompEE's high-level knowledge structures include plans, analogies, and concepts, and even its genotypic level uses hierarchies of symbolic genes for its structures. These modifications result in a system that has the characteristic advantages of a subsymbolic system that can be used in a symbolic domain.

Like the AL approach, CompEE focuses on behavior, not only on solutions. However, instead of focusing on only behavior resulting from species learning, CompEE's behavior results from the combination of species learning and individual learning. In addition to observing this joint behavior, the combination of the two enables us to look at how individual learning evolves and how it affects species learning. For example, we observe that after SimOrgs evolve condition dropping, it causes them to rely less on analogy. The benefit from this is we can determine when one or a combination of abilities is better than another for a given problem.

While most Animats use reinforcement learning architectures that are effective at only trial-and-error learning, and no more, CompEE extends the reinforcement learning architecture with schema templates and EBR, which enables CompEE to do operations that are considered cognitive, such as planning and reasoning. Our results show that a SimOrg's ability to produce schemata whose actions solve its current problems improves

the longer it uses EBR. Since its memory can not store all possible actions for given environmental conditions, a SimOrg's ability to reason from past experiences plays an important role in its survival. In domains where more than one response must be determined, a SimOrg's ability to learn from experience is critical.

CompEE's approach to intelligence which relies on evolution as the main source of information gain is a combination of many successful components from other symbolic and subsymbolic approaches. The combination overcomes many of the disadvantages associated with the individual approaches, and, by using the process of evolution for learning, adaptation is an essential and integrated part of the system, not a hand-coded addition.

CompEE's contributions are many. For traditional AI it describes a method of grounding symbols to sensory information. For CBR it describes how adaptation can become an integral component of the reasoning method rather than a hand-coded addition, where cases can be combined to form new ones, and where opportunistic, novel solutions can be constructed. For ML it identifies a method of managing the 'preprogramming' problem and a new reasoning method. For GAs it offers a technique for evolving the organization and structure of the chromosome which is critical in environments where the details of the fitness calculation are unknown. For Animats and AL, it provides an approach for the study of individual learning.

The work in EE has gone virtually unnoticed by the GA, AL and Animat communities. This is an omission that we wish to correct. EE offers many ideas on the topics of the evolution of knowledge, learning, and behavior, and should not be overlooked. Although EE has failed to advance detailed, empirically testable ideas, CompEE does just that.

8.2 Future Research Direction

Research into CompEE has opened many different issues that should be addressed by future research.

CompEE uses hierarchies to organize its genome and schemata. The reason for this is, since the specifics of the fitness computation for the genome are unknown and schemata come in different sizes, hierarchies provide the best structure for adapting their contents and organization simultaneously when combined with the appropriate operators. The results in chapter 4 demonstrate that in problems where gene ordering is critical, a GA also benefits from using hierarchies. One such problem is the Traveling Salesman Problem. However, how to map a tour onto a hierarchy is an open problem. Further study may also suggest different operators and methods of fitness evaluation for working with complete trees like those of the TGA and with subtrees like those of the STGA.

In CompEE three types of schema templates are applied to short-term memory to determine initial plans, analogies, and concepts. The results have shown that a plan template combined with the ability to perform condition dropping, or plan and analogy templates together, provides a SimOrg with the capability to produce information that enables it to function in its environment. Further study may produce new templates and functional abilities to use under genetic control. By examining which ones the species selects, one can decide which ones, or combinations, are crucial for learning and should be explored further.

EBR randomly chooses which genetic operators to apply next. Obviously, there are situations where extra checking would improve the effectiveness of the operator, e.g., making sure that the two selected schemata for the join operator have the same conditional and expected contexts. Further study into providing new operators and when they should be applied should decrease the number of times EBR makes poor adaptations.

An interesting area of further study would be to apply CompEE to environments that have been mapped from more traditional problems, such as the Traveling Salesman Problem. For example, if food sites were used to represent cities in a Traveling Salesman Problem, where the types and amounts of food at each site represents the distances to other cities, and SimOrgs survive only if they control their hunger between sites, meaning they do not travel large distances between food sites, and they are not permitted to feed from a particular food station more than once, a SimOrg could attempt to develop a tour around the food sites which solves a Traveling Salesman problem.

As it became evident when the simulation runs were done, CompEE would benefit from being implemented on a parallel machine. By its very nature, CompEE is a parallel architecture. Each SimOrg could run as a single process in a shared memory environment. Currently, they are run sequentially. However, in a parallel system, some of them could be performing actions while others are thinking. It would be interesting to see how the parallel implementation would affect their behaviors. Would they spend as much time thinking - using EBR - realizing that they could be eaten by another one at any moment?

Although our system does not create its own parameters and subsystems, it would be very interesting to see if a system could. For example, we defined three schema templates for a SimOrg to use to extract information from short-term memory. Although these are general templates, their structure was created by us, not the system. If the system could evolve its own templates and parameters for controlling them, and then make use of them, it would be a great achievement.

The crossover operators TreCoP and the STreCoP, which are used by the TGA and the STGA respectively, use weighted selection methods for determining the nodes in the trees at which crossover will occur. One method selects the root node of a tree with 50% probability, and the other selects a child of the root node with 50% probability. Although the 50% probability helped the crossover operators to constantly pull nodes toward the

root, other probabilities might provide better results. To determine these new probabilities, one could add genes into the genome that represents each probability and examine their development, or one could simply run the TGA or STGA with different combinations of probabilities and examine the results.

Finally, I believe one of the most interesting areas of future research would be to add a cultural level to CompEE that also uses evolution as the primary knowledge gaining process. Currently, a SimOrg does not receive any schemata from its parents or neighbors, and, because of this, it must learn what its parents already know. To add a cultural level, SimOrgs must be able to communicate with one another. If the sharing of information is possible, would SimOrgs be fitter? It was clear that in the plentiful environment where SimOrgs were without the senses of touch or hearing, they would have benefited from the knowledge of food that other SimOrgs had developed. Would their requirement for certain abilities, such as condition dropping, change? *Would we see the evolution of cultural knowledge?*

Appendix A

Glossary

Allele: One of several forms of the same gene, presumably differing by mutation of the DNA sequence, and capable of segregating as a unit Mendelian factor. Alleles are usually recognized by their phenotypic effects; DNA sequence variants recognized by direct sequencing are usually called haplotypes [Futuyma, 1986].

Animat: Simulated animal or autonomous robot.

Bioepistemology: The name given to Lorenz's version of Evolutionary Epistemology by Kai Hahlweg and C.A. Hooker.

Cognogenesis: The evolution of the structures and process of cognition; or, the evolution of public or communally shared knowledge [Hahlweg and Hooker, 1989b].

Darwinism: The theory of evolution formulated by Charles Darwin that holds that different species of plants and animals have arisen by a process of slow and gradual changes over successive generations brought about by natural selection [Hale & Margham, 1988].

Diploid: (of a cell nucleus) containing two of each type of chromosome in homologous pairs and formed as a result of sexual reproduction [Hale & Margham, 1988].

Epigenesis: The formation of entirely new structures during the development of the embryo [Hale & Margham, 1988].

Epigenetic: Developmental; pertaining especially to interactions among developmental processes above the level of primary gene action [Futuyma, 1986].

Epistemology: Inquiry into the nature and ground of experiences, belief and knowledge; also called the theory of knowledge [Lacey, 1990].

Gamete: A specialized haploid cell that fuses with a gamete from the opposite sex (or mating type) to form a diploid zygote [Hale & Margham, 1988].

Gene: The functional unit of heredity [Futuyma, 1986].

Genome: The complete complement of genetic material in a cell, or carried by an individual [Hale & Margham, 1988].

Genotype: The set of genes possessed by an organism; often, its genetic composition at a specific locus or set of loci singled out for discussion [Futuyma, 1986].

Haploid: (of a cell nucleus) containing one of each type of chromosome [Hale & Margham, 1988].

Instrumentalism: The belief that theories are neither true nor false, but have the status only of instruments or calculating devices for predicting the results of measurements. This amounts to the belief that the only things that are genuinely real are the results of observations [Casti, 1989].

Learning: Evolutionary changes to an individual's internal organization that improves its performance.

Neo-Darwinism: A view of evolutionary theory that combines Mendelian Genetics with Darwinism [Hale & Margham, 1988].

Ontogeny: The whole of the development of an organism from fertilization to the completion of the life history [Hale & Margham, 1988].

Phenotype: The morphological, physiological, biochemical, behavioral, and other properties of an organism, manifested throughout its life, that develop through actions of genes and environment; or any subset of such properties, especially those affected by a particular allele or other portion of the Genotype [Futuyma, 1986].

Phylogeny: The whole of the evolutionary history of a species or other taxonomic group of organisms [Hale & Margham, 1988].

Psychogenesis: The cognitive ontogenesis of the individual [Hahlweg & Hooker, 1989a].

Relativism: The belief that truth is no longer a relationship between theory and an independent reality, but rather depends at least in part on something like the social perspective of the person holding the theory. Thus as one passes from age to age, or from society to society, or from theory to theory, what's true changes [Casti, 1989].

Realism: The belief that there is an objective reality "out there" independent of ourselves. This reality exists solely by virtue of how the world is, and it is in principle discoverable by application of the methods of science [Casti, 1989].

Zygote: The diploid cell produced by the fusion of the nuclei of male and female gamete nuclei at fertilization [Hale & Margham, 1988].

Appendix B

Acronyms

AI: Artificial Intelligence.

AL: Artificial Life.

CBR: Case-Based Reasoning.

CompEE: Computational Evolutionary Epistemology.

CS: Classifier System.

EBR: Evolution-Based Reasoning.

EE: Evolutionary Epistemology.

EEl: The version of EE that attempts to account for the evolution of animal or human knowledge, throughout their lifetimes, by combining EEM with EET applied to an individual's knowledge, not scientific ideas and culture.

EEM: The version of EE that attempts to account for the evolution of the structures and processes of cognition in animals and humans by extending the theory of evolution to those aspects of animals which are the biological substrates of cognitive activity.

EET: The version of EE that attempts to account for the evolution of scientific ideas and culture using models and metaphors from evolutionary biology.

GA: Genetic Algorithm.

ML: Machine Learning.

MMA: Mill's method of agreement.

MMD: Mill's method of difference.

MJMAD: Mill's joint method of agreement and difference.

NN: Neural Network.

RGA: A Genetic Algorithm that uses random gene locations in a sequential list structured chromosome.

RP1: Riedl's first principle of behavioral constraint.

RP2: Riedl's second principle of behavioral constraint.

RP3: Riedl's third principle of behavioral constraint.

RP4: Riedl's fourth principle of behavioral constraint.

SGA: A Genetic Algorithm that use a sequential list structured chromosome.

SimOrg: Simulated Organism.

STreCoP: SubTree Crossover Operator.

STGA: A Genetic Algorithm that uses tree and subtree structured chromosomes.

TGA: A Genetic Algorithm that uses tree structured chromosomes.

TreCoP: Tree Crossover Operator.

Bibliography

- Ackley, D.H. (1987). An Empirical Study of Bit Vector Function Optimization. In Lawrence Davis (Ed.), *Genetic Algorithms and Simulated Annealing* (pp. 170-204). Los Altos, California: Morgan Kaufmann Publishing Inc.
- Ackley, D.H. & Littman, M. (1992). Interactions Between Learning and Evolution. In C. Langton, C. Taylor, J. Farmer, and S. Rasmussen (Eds.), *Artificial Life II*. (pp. 487-509). New York: Addison Wesley.
- Antonisse, J. (1989). A New Interpretation of Schema Notation that Overturns the Binary Encoding Constraint. *Proceedings of the Third International Conference on Genetic Algorithms*. San Mateo, California: Morgan Kaufmann, 86-91.
- Arthur, W. (1984). *Mechanisms of Morphological Evolution*. New York: John Wiley & Sons.
- Ayala, F.J. (1976a). *Molecular Evolution*. Sunderland, Massachusetts: Sinauer Associates.
- Ayala, F.J. (1976b). *Molecular Genetics and Evolution*. In F.J. Ayala (Ed.), *Molecular Evolution* (Ch. 1). Sunderland, Massachusetts: Sinauer Associates.
- Beer, R.D., & Chiel, H.J. (1991). Neural Basis of Behavior Choice in an Artificial Insect. From Animals to Animats: *Proceedings of the First International Conference on Simulation of Adaptive Behavior*. Cambridge, Massachusetts: The MIT Press, 247-254.
- Belew, R.K. & Booker, L.B. (1991). *Proceedings of the Fourth International Conference on Genetic Algorithms*. San Mateo, California: Morgan Kaufmann Publishers.
- Belew, R. K., McInerney, J., and Schraudolph, N. N. (1992) *Evolving Networks*. In C. Langton, C. Taylor, J. Farmer, and S. Rasmussen (Eds.), *Artificial Life II*. (pp. 511-547). New York: Addison Wesley.
- Bradie, M. (1986). Assessing Evolutionary Epistemology. *Biology and Philosophy*, 1(4), 401-459.
- Callebaut, W., & Pinxten, R. (1987). *Evolutionary Epistemology: A Multiparadigm Program*. Boston: Reidel Publishing Company.
- Campbell, Donald. T. (1987a). Blind Variation and Selective Retention in Creative Thought as in other Knowledge Processes. In G. Radnitzky and W.W. Bartley, III (Eds.), *Evolutionary Epistemology, Rationality, and the Sociology of Knowledge* (pp. 91-114). La Salle, Illinois: Open Court.

- Campbell, Donald. T. (1987b). Evolutionary Epistemology. In G. Radnitzky and W.W. Bartley, III (Eds.), *Evolutionary Epistemology, Rationality, and the Sociology of Knowledge* (pp. 47-90). La Salle, Illinois: Open Court.
- Campbell Donald. T., Heyes, C.M., & Callebaut, W. (1987). Evolutionary Epistemology Bibliography. In W. Callebaut and R. Pinxten (Eds.), *Evolutionary Epistemology: A Multiparadigm Program* (pp. 405-431). Boston: Reidel Publishing Company.
- Campbell, J.H. (1985). An Organization Interpretation of Evolution. In D.J. Depew and B.H. Wever (Eds.), *Evolution at a Crossroad: The New Biology and the New Philosophy of Science* (pp. 133-167). Cambridge, Massachusetts: MIT Press.
- Casti, J.L. (1989). *Paradigms Lost*. London: Sphere Books Ltd.
- Coderre, B. (1987). Modeling Behavior in Petworld. In C. Langton (Ed.), *Artificial Life* (pp. 407-420). New York: Addison Wesley.
- Collins, R.J. & Jefferson, D.R. (1992). AntFarm: Towards Simulated Evolution. In C. Langton, C. Taylor, J. Farmer, and S. Rasmussen (Eds.), *Artificial Life II*. (pp. 579-601). New York: Addison Wesley.
- Darwin, R.C. (1859). *The Origin of Species*. Reprinted 1985. London: Penguin Group.
- Darwin, R.C. (1969). *The Autobiography of Charles Darwin*. N. Barlow (Ed.). New York: W. W. Norton.
- Davis, L. (1987). *Genetic Algorithms and Simulated Annealing*. London: Pitman.
- Dawkins, R. (1976). *The Selfish Gene*. New York: Oxford University Press, 48.
- Dawkins, R. (1982). *The Extended Phenotype*. Oxford: W.H. Freeman.
- De Jong, K. (1990). Genetic Algorithm-based Learning. In Y. Kodratoff and R. Michalski (Eds.), *Machine Learning: An Intelligent Approach*, Vol. III (pp. 611-638). San Mateo, California: Morgan Kaufmann Publishers.
- Deugo, D.L. & Oppacher, F. (1992). Explicitly Schema-Based Genetic Algorithms. *Proceedings of the Ninth Biennial Conference of the Canadian Society for Computational Studies of Intelligence*. Palo Alto, California: Morgan Kaufmann Publishers, 46-53.
- Doolittle, W. Ford (1988). Hierarchical Approaches to Genome Evolution. In *Philosophy and Biology* (pp. 101-133). The University of Calgary Press.
- Druscher, G.L. (1991) *Made-Up Minds: A Constructivist Approach to Artificial Intelligence*. Cambridge, Massachusetts: MIT Press.
- Edelman, G.M. (1987). *Neural Darwinism: The Theory of Neuronal Group Selection*. New York : Basic Books, Inc..

- Eldredge, Niles, and Stanley N. Salthe (1984). Hierarchy and Evolution. In R. Dawkins and M. Ridley (Eds.), *Oxford Surveys in Evolutionary Biology* (pp. 184-208). Oxford University Press.
- Eysenck, M.W. & Keane, M.T. (1990). *Cognitive Psychology: A Student's Handbook*. Hillsdale: Lawrence Erlbaum Associates.
- Fogelmann-Soulie, F., Gallinari, P., and Thiria, S. (1990). *Network Learning*. unpublished manuscript.
- Futuyma, D.J. (1986). *Evolutionary Biology*. Sunderland, Massachusetts: Sinauer Associates.
- Gabbert, P.S., Brown, D.E., Huntley, C.L., Markowicz, B.P., and Sappington, D.E. (1998). A System for Learning Routes and Schedules with Genetic Algorithm. *Proceedings of the fourth International Conference on Genetic Algorithms*. San Mateo, California: Morgan Kaufmann, 430-436.
- Goldberg, D.E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. New York: Addison-Wesley.
- Grant, V. (1981). *Plant Speciation*. 2nd edn. New York: Columbia University Press.
- Grefenstette, J.J. (1988). Credit Assignment in Genotypic learning Systems. *Proceedings of the Seventh National Conference on Artificial Intelligence*. San Mateo, California: Morgan Kaufmann Publishers, 596-600.
- Grefenstette, J.J. (1989). A System for Learning Control Strategies with Genetic Algorithms. *Proceedings of the Third International Conference on Genetic Algorithms*. San Mateo, California: Morgan Kaufmann, 183-190.
- Grefenstette, J.J. (1992). The Evolution of Strategies for Multiagent Environments. *Adaptive Behavior* 1(1). Cambridge, Massachusetts: MIT Press, 65-90.
- Guralnik, D.B. (1983). *Webster's New World Dictionary*. New York: Warner Books.
- Hahlweg, K., & Hooker, C.A. (1989a). *Issues in Evolutionary Epistemology*. Albany: State University of New York Press.
- Hahlweg, K., & Hooker, C.A. (1989b). Evolutionary Epistemology and Philosophy of Science. In K. Hahlweg & C.A. Hooker (Eds.), *Issues in Evolutionary Epistemology* (pp. 21-150). Albany: State University of New York Press.
- Hale, W.G., & Margham, J.P. (1988). *Collins Dictionary of Biology*. London: HarperCollins.
- Hammond, K.J. (1989). *Case-Based Planning: Viewing Planning as a Memory Task*. Boston: Academic Press, Inc..

- Hinton, G.E. (1990a) Connectionist Learning Procedures. In J. Carbonell (Ed.) Machine Learning, Paradigms and Methods (pp. 185-234). Cambridge, Massachusetts: MIT Press.
- Hinton, G.E. (1990b) Connectionist Learning Procedures. In Y. Kodratoff and R. Michalski (Eds.), Machine Learning: An Intelligent Approach, Vol. III (pp. 555-610). San Mateo, California: Morgan Kaufmann Publishers.
- Holland, J.H. (1975). Adaptation in Natural and Artificial Systems. Ann Arbor: University of Michigan Press.
- Holland, J.H. (1986). Escaping Brittleness: The Possibilities of General-Purpose Learning Algorithms Applied to Parallel Rule-Based Systems. In Ryszard S. Michalski, J. G. Carbonell, and T. M. Mitchell (Eds.), Machine Learning: An Artificial Intelligence Approach, Vol. II (pp. 593-634). San Mateo, California: Morgan Kaufmann Publishers.
- Holland, J.H., Holyoak, K.H., Nisbett, R.E., and Thagard, P.R. (1986). INDUCTION: Processes of Inference, Learning, and Discovery. Cambridge, Massachusetts: MIT Press.
- Hull, D. (1981). Units of Evolution: A Metaphysical Essay. In U.L. Jensen and R. Harre (Eds.), Philosophy of Evolution (pp. 23-44). Harvester Press.
- Hull, D. (1988). Interactors versus Vehicles. In H.C. Plotkin (Ed.), The Role of Behavior in Evolution (pp. 19-50). Cambridge, Massachusetts: MIT Press.
- Hunkapiller, T., Henry Huang, Leroy Hood, and John H. Campbell (1982). The Impact of Modern Genetics on Evolution Theory. In Roger Milkman (Ed.), Perspectives On Evolution (pp. 64-189). Sunderland, Massachusetts: Sinauer Associates Inc..
- Judd, J.S. (1990). Neural Network Design and the Complexity of Learning. Cambridge, Massachusetts: The MIT Press.
- Kant, I. (1781). Kritik der reinen Vernunft. Riga, Latvia: Hartknoch. English translation, London: Bell.
- Knuth, D.E. (1973). The Art of Computer Programming: Fundamental Algorithms, V:1. Menlo Park, California: Addison-Wesley Publishing Company, 391.
- Kolodner, J.L., Simpson, K., and Sycara-Cryanski, K. (1987). A Process Model of Case-Based Reasoning in Problem Solving. Proceedings of the Sixth National Conference of Artificial Intelligence. San Mateo, California: Morgan Kaufmann Publishers, 284-290.
- Kolodner, J.L.(1988). Extending Problem Solver Capabilities Through Case-Based Inference. Proceedings: Case-Based Reasoning Workshop, San Mateo, California: Morgan Kaufmann Publishers, 21-30.

- Kordig, C.R. (1982). Evolutionary Epistemology is Self-Referentially Inconsistent. *Philosophy and Phenomenological Research*, 42, 449-450.
- Koza, J. (1992). Genetic Evolution and Co-Evolution of Computer Programs. In Christopher Langton, Charles Taylor, J. Doyne Farmer, and Steen Rasmussen (Eds.), *Artificial Life II* (pp. 603-630). New York: Addison-Wesley.
- Lacey, A.R. (1990). *A Dictionary of Philosophy*. New York: Routledge.
- Langton, C. (1989) *Artificial Life*. New York: Addison-Wesley.
- Langton, C., Taylor, C., Farmer, J., and Rasmussen, S. (1992) *Artificial Life II*. New York: Addison-Wesley.
- Lorenz, K. (1941). Kant's Doctrine of A Priori in Light of Contemporary Biology. English translation in H.C. Plotkin (Ed.), *Learning, Development, and Culture: Essays in Evolutionary Epistemology* (pp., 121-143). Chichester, England: Wiley.
- Lorenz, K. (1973). *Behind the Mirror*. New York: Harcourt Brace Jovanovich.
- Løvtrup, S. (1974). *Epigenetics*. New York: John Wiley & Sons.
- Lumsden, C.J. & Wilson, E.O. (1981). *Genes, Mind, and Culture: The Coevolutionary Process*. Cambridge, Mass: Harvard University Press.
- Mendel, G. (1981). Experiments in Plant Hybridization. (Translation.) In C.I. Davern (Ed.), *Genetics, A Scientific American Reader*. Oxford: W. H. Freeman.
- Meyer, J.A. & Wilson, S.W. (1991). From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior. Cambridge, Massachusetts: The MIT Press.
- Michalski, R. & Kodratoff, Y. (1990). Research in Machine Learning. In Y. Kodratoff and R. Michalski (Eds.), *Machine Learning: An Intelligent Approach*, Vol. III (pp. 3-30). San Mateo, California: Morgan Kaufmann Publishers.
- Mill, J.S. (1911). *A System of Logic*. New York: Longmans, Green, and Co..
- Nilsson, N.J. (1980). *Principles of Artificial Intelligence*. Palo Alto, California: Tioga Publishing Company.
- Oppacher, F. & Deugo, D.L. (1991) Integrating Case-Based Reasoning with Genetic Algorithms. In N. Cercone and G. Gardin (Eds.) *Computational Intelligence III* (pp. 103-114). North-Holland.
- Pattee, H.H. (1973). The Physical Basis and Origin of Hierarchical Control. In H.H. Pattee (Ed.), *Hierarchy Theory* (pp. 71-108). New York: Braziller.
- Piaget, J. (1970). *Genetic Epistemology*. New York: Columbia University Press.

- Plotkin, H.C. (1986). An Evolutionary Epistemological Approach to the Evolution of Intelligence, In Harry J. Jerison and Irene Jerison (Eds.), *Intelligence and Evolutionary Biology* (pp. 73-92). Springer-Verlag.
- Popper, K.R. (1972). *Objective Knowledge: An Evolutionary Approach*. Oxford, England: Clarendon Press.
- Popper, K.R. (1974). Replies to My Critics. In P. A. Schilpp (Ed.), *The Philosophy of Karl Popper*, La Salle: Open Court, 1061.
- Popper, K.R. (1987). Natural Selection and the Emergence of Mind. In G. Radnitzky and W.W. Bartley, III (Eds.), *Evolutionary Epistemology, Rationality, and the Sociology of Knowledge* (pp. 139-156). La Salle, Illinois: Open Court.
- Quine, W.V.O. (1971). *Ontological Relativity and Other Essays*. New York: Columbia University Press.
- Radnitzky, G., & Bartley, W.W., III (1987). *Evolutionary Epistemology, Rationality, and the Sociology of Knowledge*. La Salle, Illinois: Open Court.
- Riedl, R. (1980). Evolution and Evolutionary Knowledge. In F.M. Wuketits (Ed.), *Concepts and Approaches in Evolutionary Epistemology* (pp. 35-50). Boston: D. Reidel Publishing Company.
- Riedl, R. (1984). *Biology of Knowledge*. New York: Wiley.
- Riesbeck, C.K. & Schank, R.C. (1989) *Inside Case-Based Reasoning*. Hillsdale, New Jersey: Lawrence Erlbaum Associates.
- Rizki, M.M. & Conrad, M. (1986). Computing The Theory of Evolution. *Physica D*, **22**, 83-89.
- Ross, B. H. (1989). Some Psychological Results on Case-Based Reasoning. *Proceedings: Case-Based Reasoning Workshop*. San Mateo, California: Morgan Kaufmann Publishers, 144-147.
- Ruse, M. (1986). *Taking Darwin Seriously*. New York: Basil Blackwell.
- Schaffer, J.D. (1989). *Proceedings of the Third International Conference on Genetic Algorithms*. San Mateo, California: Morgan Kaufmann Publishers.
- Schank, R. (1986). *Explanation Patterns*. Lawrence Erlbaum Associates.
- Schilpp, P.A. (1974). *The Philosophy of Karl Popper*. LaSalle, Illinois: Open Court.
- Schnepf, Uwe (1991). *Robot Ethology: A Proposal for the Research Into Intelligent Autonomous Systems. From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior*. Cambridge, Massachusetts: The MIT Press, 465-474.

- Simon, Herbert A. (1969). *The Sciences of the Artificial*. Cambridge, Massachusetts: The MIT Press.
- Simon, Herbert, A. (1973). *The Organization of Complex Systems*. In H.H. Pattee (Ed.), *Hierarchy Theory* (pp. 1-28). New York: Braziller.
- Skyrms, Brian (1975). *Mill's Methods of Experimental Inquiry and the Nature of Causality*. In *Choice and Chance: An Introduction to Inductive Logic* (Ch 4). Dickenson Publishing Company.
- Sutton, S.S. (1991). *Reinforcement Learning Architectures for Animats*. From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior. Cambridge, Massachusetts: The MIT Press, 288-296.
- Spencer, H. (1870). *The Principles of Psychology*. London: Murray.
- Syswerda, G. (1989). *Uniform Crossover in Genetic Algorithms*. Proceedings of the Third International Conference on Genetic Algorithms. San Mateo, California: Morgan Kaufmann Publishers, 2-9.
- Thagard, P. (1988) *Computational Philosophy of Science*. Cambridge, Massachusetts: The MIT Press.
- Thompson, P. (1988). *Conceptual and Logical Aspects of the 'New' Evolutionary Epistemology*. *Canadian Journal of Philosophy* 14. Calgary: The University of Calgary Press, 235-253.
- Thorndike (1987). In Robert J. Richards (Ed.), *Darwin and The Emergence of Evolutionary Theories of Mind and Behavior*. Chicago: University of Chicago Press, 440.
- Toates, F. (1986). *Motivation Systems*. Cambridge: Cambridge University Press.
- Toulmin, S. (1981). *Evolution, Adaptation, and Human Understanding*. In M.B. Brewer & B.E. Collins (Eds.), *Scientific Inquiry and the Social Sciences: A Volume in Honor of Donald T. Campbell* (pp. 18-36). San Francisco: Jossey-Bass.
- Travers, M. (1987). *Animal Construction Kits*. In C. Langton (Ed.), *Artificial Life* (pp. 421-440). New York: Addison Wesley.
- Tyrrell, T., & Mayhew J.E.W. (1991). *Computer Simulation of an Animal Environment*. From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior. Cambridge, Massachusetts: The MIT Press, 263-272.
- Vollmer, G. (1987). *On Supposed Circularities in an Empirically Oriented Epistemology*. In G. Radnitzky and W.W. Bartley, III (Eds.), *Evolutionary Epistemology, Rationality, and the Sociology of Knowledge* (pp. 163-200). La Salle, Illinois: Open Court.

- Wilson, E.O. & Bossert, W.H. (1971). *A Primer of Population Biology*. Sunderland, Massachusetts: Sinauer Associates.
- Wilson, S. W. (1989). The Genetic Algorithm and Simulated Evolution. In C. G. Langton (Ed.), *Artificial Life* (pp. 157-166). New York: Addison-Wesley.
- Wilson, S. W. (1991). The Animate Path to AI. From Animals to Animats: Proceedings of the First International Conference on Simulation of Adaptive Behavior. Cambridge, Massachusetts: The MIT Press, 15-21.
- Winograd, T. (1983). *Language as a Cognitive Process*. New York: Addison Wesley.
- Wuketits, F.M. (1984a). *Concepts and Approaches in Evolutionary Epistemology*. Dordrecht, Holland: D. Reidel Publishing Company.
- Wuketits, F.M. (1984b). Evolutionary Epistemology - A Challenge to Science and Philosophy. In F.M. Wuketits (Ed.) *Concepts and Approaches in Evolutionary Epistemology* (pp. 1-34). Dordrecht, Holland: D. Reidel Publishing Company.
- Wuketits, F.M. (1986). Evolution as a Cognition Process: Towards an Evolutionary Epistemology. *Biology and Philosophy* 1:2. Dordrecht, Holland: D. Reidel Publishing Company, 191-206.
- Wuketits, F.M. (1990). *Evolutionary Epistemology and Its Implications for Humankind*. New York: State University of New York Press.

END

1 | 9-07-94

FIN