INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

ProQuest Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

UMI
A Complete Implementation of John Holland’s
Echo Model for Complex Adaptive Systems

By Lee Graham

A thesis submitted to the Faculty of Graduate Studies and
Research in partial fulfillment of the requirements for the
degree of:
Master of Computer Science

Carleton University
August 15, 2001

© Copyright
2001, Lee Graham
The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L’auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

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

0-612-66870-3
The undersigned recommend to the Faculty of Graduate Studies and Research the acceptance of the thesis

"A Complete Implementation of John Holland’s Echo Model for Complex Adaptive Systems"

submitted by Lee Graham in partial fulfillment of the requirements for the degree of Master of Computer Science.

Director, School of Computer Science

Thesis Supervisor

Carleton University
August, 2001
Abstract

Many natural and man-made systems exhibit high-level emergent behaviors, which result from numerous intricate interactions within a large population of primitive evolving components. Such systems are known as "Complex Adaptive Systems" or "CAS" and are extremely difficult to model using conventional modeling techniques. John Holland, of the Santa Fe Institute in New Mexico, has described a class of abstract agent-based CAS models known as "Echo". This project describes a complete working implementation of an Echo model, the design decisions required to create it and some preliminary testing of the system.
Acknowledgments

I’d like to thank my supervisor and the members of ALEC for their input, including useful comments, suggestions and corrections regarding terminology. I’d also like to thank the members of the DebunkCreation list at Yahoo! Groups who provided help by pointing me toward information on the evolution of metazoans. This research is funded in part by the Natural Sciences and Engineering Research Council of Canada.
Table of Contents

Abstract iii

Acknowledgments iv

Table of Contents v

Table of Figures viii

1. Introduction 1

1-1 Contributions 1

1-2 What is ALife? 2

1-3 What is a Complex Adaptive System (CAS)? 4

1-4 What is Holland's Echo model? 6

2. Holland's Echo model 12

2-1 The Echo world 13

2-2 Resources 14

2-3 Level 1: Offence, defense and agent reservoirs 15

2-4 Level 2: Conditional exchange 18

2-5 Level 3: Resource transformation 20

2-6 Level 4: Agent boundaries and adhesion 22

2-7 Level 5: Selective mating 27

2-8 Level 6: Conditional replication and multiagents 28

2-9 Previous Work 32

3. An Echo Implementation 35

3-1 The Echo World class 36

3-1-1 Dimension-like parameters 36
3-1-2 Probability parameters 39
3-1-3 World statistics 43
3-1-4 Gene offsets and other related variables 45
3-1-5 Logging 46
3-2 The Site class 47
3-3 The Agent class 50
3-4 The Boundary class 54
3-5 Genome tracking 55
3-6 Offence, defense and resource absorption 59
3-7 Conditional exchange 63
3-8 The enzyme gene 65
3-9 Boundary configurations and issues 66
3-10 Mating and crossover 71

4. Testing 75
4-1 Trade 75
4-2 Cooperation 78
4-3 Dominance through Differential Reproduction 79
4-4 Extinction 80
4-5 Predation by combat 81
4-6 Predation by Trade 83

5. Conclusion 84
5-1 What could have been done better? 84
5-2 Future work 85
# Table of Figures

**Figure 1**: An overview of the Echo world

**Figure 2**: Interaction in level 1

**Figure 3**: Determining the outcome in trade interactions

**Figure 4**: Interaction in level 2

**Figure 5**: Example of an enzyme gene

**Figure 6**: Interaction in level 4

**Figure 7**: Daughter agent is ejected from aggregate

**Figure 8**: Daughter agent shares boundary of mother

**Figure 9**: Daughter agent is engulfed by mother

**Figure 10**: An aggregate and its corresponding tree representation

**Figure 11**: An agent's domain of interaction

**Figure 12**: Interaction in level 5

**Figure 13**: Illustration of multiagent reproduction

**Figure 14**: Number of resources vs. number of stagnant runs

**Figure 15**: Population graph of a 2000-cycle run

**Figure 16**: Stacked population graph from a 2000-cycle run

**Figure 17**: Use of barrier sites to alter world geography

**Figure 18**: World view showing trade interactions

**Figure 19**: Claimed vs. surplus resources in a multiagent

**Figure 20**: Example of a trie tree (29 nodes)

**Figure 21**: Example of a trie tree with path compression (9 nodes)
Figure 22: Frequencies of replication factor values
Figure 23: Two uptake genes
Figure 24: Roulette wheel for resource distribution
Figure 25: 10 examples of uptake and trading genes
Figure 26: Breakdown of agent interaction
Figure 27: Boundary "bursting"
Figure 28: Boundary "emptying"
Figure 29: Agents that eject their daughters
Figure 30: Agents that engulf their daughters
Figure 31: Agents whose daughters share their boundaries
Figure 32: Stacked population graph of a typical 500-cycle run
Figure 33: A 500-cycle run without reducing the chance of mating
Figure 34: Trade example 1
Figure 35: Trade example 2
Figure 36: Trade example 3
Figure 37: Cooperation example 1
Figure 38: Cooperation example 2
Figure 39: Differential Reproduction example
Figure 40: Extinction example
Figure 41: Predation by Combat example 1
Figure 42: Predation by Combat example 2
Figure 43: Predation by Combat example 3
Figure 44: Predation by Trade example
1. Introduction

1-1 Contributions

The primary contributions of this work are as follows:

- As of the date of this writing, to the best of my knowledge and that of my supervisor, the specific Echo model described herein marks the first such implementation to include all six levels as described by John Holland, and thus embodies the first complete realization of Echo.

- This document represents a discussion and rationalization of the numerous, sometimes arbitrary, design decisions that a would-be implementer of Holland's model will be faced with.

- Since this is a full implementation of Echo, it is now possible to conduct an extensive range of CAS related experiments relating to such things as the evolution of multicellular organisms, evolutionary dynamics, embryogenesis and other phenomena of potential interest to biology and other CAS-related fields.
1-2 What is ALife?

ALife is the simulation of life in an artificial medium and the study of artificial models of biological phenomena. The term describes research carried out using systems designed to simulate essential properties of living organisms. Much of the work done in the field of ALife revolves around the study of life as we find it here on earth, but there is also significant effort to study principles and properties of living systems in a more general sense, not just the carbon-based versions of life we find here on earth. Dr. Christopher Langton, editor of the journal "Artificial Life" and research scientist at the Santa Fe Institute in New Mexico, is generally recognized as the founder of the ALife field. While working as a research fellow at the Los Alamos National Laboratory in 1987, he organized the first ALife conference.

"Among all of the things that artificial life is or will come to be, however, it is probably safe to say that the field as a whole represents an attempt to vastly increase the role of synthesis in the study of biological phenomena."

If one accepts the idea that a particular chemistry or substrate is not a fundamental part of what constitutes life then life becomes a much more general term and what was once considered simulation now takes a step closer to being alive. Much in the way that a virus lies in the conceptual gradient between molecules and living organisms, to some, ALife lies in the gap between simulation and the real McCoy. There are a weak and a strong claim about computer models of life. The weak claim is that such systems are merely models of the real thing. The strong claim is that the hardware of life is irrelevant
to its definition and that what is important are the behaviors, patterns and relationships of
the parts; that life can exist in artificial media. In ALife, the term “artificial” refers not to
the life, but rather to medium in which it “lives”.

A great many ALife models and systems involve an evolving population of entities
within a virtual environment. Generally, the problem these entities endeavor to solve is
that of survival, the ability to have one’s genetic material passed on into future
generations. This is very similar to other examples of evolutionary computing but with
one noteworthy difference. The fitness of an individual in an ALife population is a result
of interactions with the virtual environment, its neighbors and other constituents of the
system including but not limited to the genetic code of the individual. Such a fitness
function is said to be implicit or endogenous to the system. Fitness in such cases is not a
measure directly imposed by the user or designer, but is a dynamic and usually
unpredictable function. This is in contrast to the typical genetic algorithm in which the
fitness function is directly imposed by the user or designer. Such fitness functions are
usually static, depend only (or at least primarily) on the genetic code of the individual in
the population and are referred to as explicit or exogenous fitness functions. This
distinction is noteworthy since it places ALife models conceptually closer to biological
evolution than other examples of evolutionary computing. Also, in systems with implicit
fitness functions, due to their more dynamic nature, it is often meaningless to make
reference to some optimum solution or individual. The peaks of the fitness landscapes in
ALife systems can become troughs and valleys as the virtual environment changes. This
makes ALife models less susceptible to convergence and more apt to produce perpetual novelty.

1-3 What is a Complex Adaptive System (CAS)?

Many systems, such as ecologies, immune systems and societies, to name a few examples, are difficult to describe, predict or control with traditional methods. A major part of the difficulty with such systems is that the interactions among system components are nonlinear. As such systems become larger, behaviors emerge at higher levels in the system which are unintended or unanticipated. Systems with interesting emergent properties and behaviors resulting from nonlinearities in the interactions of their components are often called “complex systems”. When the low-level components and interactions of complex systems are able to evolve as the system progresses, these systems are then called “complex adaptive systems”, or CAS for short.

Complex adaptive systems have the following properties:

- High-level order is emergent from low-level interactions and not predetermined
- The system consists of a large population of low-level components
- These components interact with each other and their environment
- These components also evolve over time to modify their behavior
The low-level components of a CAS are referred to as "agents". Agents seek to maximize some fitness value or related measure. In an economy this may be the value of a stock portfolio or the profit of a small business, in biological evolution this may simply be survival and reproduction. Agents develop schema which are rules governing their interactions. These schema are often numerous in a given system and compete with each other via the success of the agents. It is difficult to separate phenomena in CAS from their surrounding environment and from other agent and schema populations present. Often very small low-level changes in the system can have drastic global effects. The same goes for small changes to system parameters or initial starting conditions.

There is a drawback to the agent-based approach to CAS modeling. Most modeling theories use a correspondence between parts of the system being modeled to the parts of the model. In agent-based CAS models there is a deliberate attempt to brush away the intricate details of any specific system in favor of a very abstract model that displays emergent behavior similar to the real system. For this reason it is often difficult or impossible to create a mapping to real world objects with the same clarity and usefulness of normal modeling techniques. In conventional models, the modeling system is used to make quantitative predictions about the behavior of the real world or to determine optimum values for particular variables and parameters by running a simulated experiment. Such quantitative predictions are usually not possible with agent-based CAS models and the focus must be diverted to identifying trends, behaviors or situations in a much more general sense.
A large number of naturally occurring systems like ecologies, economies, brains, immune systems, societies... etc and an increasing number of man-made systems like parallel and distributed computing systems, neural networks, AI applications and evolutionary computing systems, for example, seem to exhibit complex behavior that is emergent from the nonlinear interactions among a large number of primitive component systems and hence, qualify as CAS. For this reason, researchers in many fields such as computer science, complexity theory, biology, cognitive science, economics, mathematics, neuroscience... etc have begun to do research and study of CAS to gain a better understanding of these systems.

1-4 What is Holland's Echo model?

A particular class of agent-based CAS models called “Echo” models is outlined in the book “Hidden Order: How Adaptation Builds Complexity” by John Holland [Holland]. The book is a collection of talks given at the 1994 Stanislaw M. Ulam memorial lectures given at the Santa Fe Institute in Santa Fe, New Mexico. Echo is intended to capture generic properties and behaviors of CAS in an agent-based model but is not meant to model any particular CAS in detail, though biological evolution seems to have been a significant influence in Holland’s conceptualization of this class of models. Holland’s model is developed gradually in six levels, some of which build upon lower levels. It is possible, to some extent, to pick and choose which levels to implement. The specific implementation discussed in sections 3-1 and beyond includes all six Echo levels.
Components and interactions in Echo are very abstract and it isn’t yet known how well it models real world phenomena.

There are two noteworthy ways in which Echo differs from standard genetic algorithms:

- As mentioned earlier, fitness in Echo is implicit, it comes from within the system
- Individuals in the population have not only a genetic code, but also a dynamic state which changes through time

The agents in an Echo model inhabit and evolve in a structured world. This world is composed of a two-dimensional lattice of “sites”. Agents inhabit one site at any given time, but can migrate through their world by moving from one site to a neighboring site. Each site can contain many agents at a time. Agents consume, trade and are themselves composed of units of renewable resources that are produced at each site at rates that may differ from site to site. These resources have no particular correspondence to quantities or items in any real world CAS, but are used in whatever manner happens to evolve in a given run.
Agents in an Echo model interact in three primary ways:

- Trading (which becomes combat when taken to an extreme)
- Mating (in which there is a sharing of genetic material)
- Adhesion (in which agents can aggregate into hierarchical structures)

Mutation during asexual reproduction, mating and selection pressure through resource constraints, interactions and a kind of ‘metabolic’ taxation provide the necessary ingredients for evolution within the system. Holland’s description of the Echo class of models is given more detailed coverage below in section 2-1 and beyond.

There are many known CAS phenomena, which Holland hopes to capture within the behavior of an Echo model. One example of this is something he refers to as the “lever-point phenomenon”. A lever-point is a disturbance in the system, usually relatively minor, that can have drastic global effects. Examples of this include such things as vaccines, which can mean the difference between life and death in an immune system, certain kinds of sudden shock or fright, which can permanently affect one’s nervous system, the introduction of an organism into an environment in which it has no natural enemy, and so on. His initial motivation for creating Echo came from a request, given to him by Murray Gell-Mann, to create a simple model of the evolution of complex structures through natural selection.
Holland also hoped to incorporate in his model such phenomena as arms races, in which two or more competing entities or groups must continually adapt to keep up with their "opponents". Mimicry, in which populations evolve to resemble another population for the purposes of concealment or protection from predation, is another such phenomenon. Other examples in the same vein include such things as the prisoner's dilemma, scavenging, co-evolution and an analog of antigen-antibody matching.

1-5 The connection to biological evolution

As was mentioned in section 1-3, biological evolution has clearly been the CAS that influenced Holland's ideas to the greatest extent in designing Echo. In order to better understand and think about an abstract CAS model such as Echo, it is worth outlining some of the similarities to a real world counterpart with which it seems to share the most direct mapping. Biological evolution best serves this purpose in discussing both the design of Echo and the terminology Holland used to describe it.

Holland provides his agents with a genetic code, which he refers to as a "chromosome". At first, the term "chromosome" seems inappropriate since biological organisms have genomes composed of multiple chromosomes whereas agents possess only one. However, in an Echo model in which all six levels are implemented, agents are given the ability to aggregate into larger entities called "multiagents", which posses multiple chromosomes in their genome.
Agents in Echo can exchange genetic material with one-another, creating hybrid agents. This is accomplished with a crossover operation on the agents’ chromosomes and this interaction is referred to as “mating”, described in section 2-7. This type of genetic swapping is akin to the way in which some bacteria are known to exchange genetic material.

Another example in which Holland uses terminology taken from biology occurs in the third level of the Echo model, described in section 2-5. In this level agents are endowed with the ability to transform resources of one type into another type at some fixed cost, allowing agents to compensate for resources that may be in short supply in their environment. Holland employs ideas from cellular biology to explain this operation, likening it to a rearrangement of the “molecular structure” of a resource to transform it into another, giving agents some abstract counterpart to a chemical enzyme.

Another aspect of Echo’s design relating to biological evolution is Holland’s attempt to capture, in an abstract way, the process of metazoan embryogenesis. Metazoans are multicellular animals with cells that are differentiated into tissues, organs, organ systems... etc. Embryogenesis is the process by which a single cell develops into a complete multicellular organism. In metazoans each cell in the body contains the same genetic material, with minor exceptions such as sperm and egg cells, and yet these cells exhibit a range of specialized behaviors and forms such as muscle, brain, fat, lung, bone... etc. Though the actual details are intricate and complex, at the heart of this increasing organization and differentiation during development is the switching on and
off ("derepression" and "repression", respectively) of genes in a different way within each cell type. Within Echo, Holland hoped to construct a simple model of this process. By first allowing a kind of agent aggregation, Echo models a phenomenon similar to slime molds where a group of cells unite into a single entity but where there is no cell differentiation. Through a process, which Holland calls "conditional replication", described in section 2-8, multiagents are granted the ability to repress and derepress genes, allowing the potential evolution of embryogenesis.

One final similarity is worth noting. Holland's agent adhesion interaction, whereby it is possible in some circumstances for one agent to be "engulfed" by another, is reminiscent of Lynn Margulis' Endosymbiotic Theory of Eukaryote Evolution [Margulis] in which bacterial ancestors of organelles like chloroplasts and mitochondria became symbiotic inhabitants of ancient prokaryotic host cells.
2. Holland’s Echo model

This section takes a closer look at Echo as it was described in “Hidden Order”. Much of the specifics of the model were left open and unspecified, making Echo a class of models as opposed to a specific instance. Due to the ambiguity and looseness of Holland’s description of Echo, it seems reasonable that the implementation details which I have made that are above and beyond the model, or those which run contrary to it, still nonetheless place my own realized instance within the class of Echo models. Much of the implementation details given by Holland are taken not as rules etched in stone but rather as suggestions. They are only one possibility out of many. My own implementation, described in section 3-1 and beyond, though different in ways from the model described in “Hidden Order” (and in the following sections) is still an Echo model.
2-1 The Echo world

The virtual habitat or world in an Echo model consists of a lattice of interconnected sites. Usually this is depicted as a two-dimensional array of sites connected to one another in the north, south, east and west directions, although this is not specified by Holland. The arrangement and interconnection of sites is arbitrary.

Agents are free to migrate from one site to an adjacent site and each site may contain many agents. Sites may also contain entities called aggregates and similar entities called multiagents, which are both hierarchical organizations of agents, which, like lone agents, migrate from site to site. All agents within a particular aggregate or multiagent migrate together and are never distributed over multiple sites.

![Figure 1: An overview of the Echo world](image)

Agents in Echo are composed of a string of resources (see section 2-2) called a "chromosome" which serves as their genetic material, and have an internal state in the form of a reservoir of resources. Agents are also assigned to a particular boundary, which can be thought of as a kind of membrane or enveloping sheath. Each agent is assigned to only one boundary but a boundary may harbor many agents (see section 2-6).
2-2 Resources

Resources are the "stuff" of the Echo world. There are usually a small number of these, perhaps four or five, designated by the letters "a", "b", "c" and so on. These resources are the food, water and sunlight of the agents, though no resource is meant to represent any particular quantity, item or substance in any given real world CAS. The "meaning" or "purpose" for a particular type of resource is simply left open for evolution to determine as it sees fit. There are no properties associated with each resource type other than the idea that two resources of a given type are, for all intents and purposes, identical and two resources of different types are not.

Each site in the world produces a subset of the existing resource types. Holland refers to this as a "fountain" or "upwelling" of resources. The particular subset of resources and the rate at which they are each produced is a property of the site and may differ from one site to another. Thus, the world may consist of some areas that are rich in many resources, like a rainforest or a swamp, and other areas that produce no resources or perhaps a single resource, like a desert or arctic tundra. Agents should be capable of evolving to adapt to these different environments. These resources are absorbed by the agents present at a site and are placed in their reservoirs. The chromosomes of agents and multiagents in Echo are constructed from strings of these very same resources. This is a very elegant aspect of Echo, since agents reproduce asexually by gathering enough resources to make a duplicate copy of the resource string specified by their chromosome.
2-3 Level 1: Offence, defense and agent reservoirs

Holland's agents are composed of two parts. The first is a string of resource letters comprising the chromosome. The second is a reservoir for storing resources, which the agents obtain in a variety of ways. In this first level of Echo, which Holland designates as "Model 1", the chromosome is divided into only two parts called the "offence tag" and the "defense tag". These tags mediate all agent interactions. The idea is to pair specific tags with one-another during an encounter between two agents, calculate a score based on string matching and use this score to determine the outcome of agent interactions. Tags do not have a fixed length and may even be empty strings in some cases. Holland suggests that one resource letter be designated as a kind of wild card or "don't care" symbol when it comes to string matching. Match scores are to be calculated as follows:

- Tags are aligned together on their leftmost letters
- The wild card letter matches any other letter
- Matching letter pairs increase the score by some predetermined amount
- Differing letter pairs decrease the score by some predetermined amount
- When one tag is longer than the other, each unpaired letter counts for some predetermined amount (positive or negative)
There exists, in this first level, only a trading interaction between agents. In each cycle of the simulation, agents on a given site are paired together and the offence tag of each is matched against the defense tag of the other.

![Diagram of agent interaction](image)

**Figure 2: Interaction in level 1**

The amount of resources transferred from one agent to its neighbor's reservoir, in this interaction, depends on the match score between its defense tag and its neighbor's offence tag. Depending on the magnitude of the match score, an agent may be required to transfer more resources than are available in its reservoir. In such cases the agent is made to transfer resources that are held in its chromosome. This is a fatal event for an agent.
Holland suggests a scoring system similar to that depicted below.

![Diagram showing trade interactions](image)

*resources beyond those needed to duplicate the chromosome

**Figure 3: Determining the outcome in trade interactions**

The primary role of the reservoir is to collect enough resources to make a duplicate of the agent's chromosome string. Resources are collected via the trade interaction described above and by absorbing them from the fountains located at each site. Once an agent has collected enough resources to copy its chromosome, it reproduces asexually, with a small chance of mutation. Mutations may lengthen or shorten a particular tag or they may simply replace one letter in the chromosome string with some other letter. In this way, fitness in level 1 is implicit in the agents' ability to collect resources. This, in turn, arises from the agents' success during trade interactions with those neighbors they happen to come in contact with, as well as the subset of resources produced in visited sites and their rates of production. Already, the factors contributing to the success of a genome are complex. With each new level of the model, this complexity will be enriched yet further.
One example, which Holland uses as an illustration of the kinds of relationships that can arise within this basic model, is that of a three-way relationship between a certain species of ant, fly and caterpillar. The fly preys on the caterpillar by laying eggs on it. The resulting larva then feed on the caterpillar. The ant, however, preys on the fly, but does not prey on the caterpillar; rather, the caterpillar secretes nectar from its skin, which the ant consumes. This relationship is stable until one element is removed. Holland depicts a similar relationship in Echo in the following way. The "caterpillar" is an agent with offence tag "c" and defense tag "aab", the "fly" has tags "aab" and "aaa", and the "ant"'s tags are "aaa" and "cb". Since the ant's offence tag matches the fly's defense tag, it preys on the fly. The fly's offence tag matches the caterpillar's defense tag, allowing it to prey on the caterpillar, whereas the interaction between the ant and caterpillar results in trade instead of combat.

One final point to mention is that agent migration occurs due to "hunger". If an agent does not successfully obtain resources, either through absorption or trade, that agent may travel to a neighboring site. Thus, agents will not remain in an area in which they are fairing poorly, but will wander until they find a more suitable location.

2-4 Level 2: Conditional exchange

This first extension to the basic model laid out in the previous section provides agents with the possibility of rejecting trade interactions. This is done with the addition of a new tag-mediated interaction, which occurs prior to trading.
The agent chromosome is now divided into two regions, a tag region, containing the offence and defense tags of level 1, and a "control" region, containing a new gene called the "exchange condition". It is at this point that Holland restricts his definition of an agent tag in order to draw his desired distinction between tags and conditions. Tags, in this level of Echo, are now restricted to a subset of the original resource alphabet. All letters are now legal in composing a tag, except for the designated wild card resource. For example, if the resource alphabet consists of [a, b, c, d] and letter "d" has been designated as the wild card, tags are restricted to the subset [a, b, c] whereas conditions may use the complete alphabet. Thus, conditions, in this level and all subsequent levels, operate identically to the tags of level 1, whereas tags have simply become more restricted in form. Another way to view this distinction is to think of tags as being open for inspection by other agents during an interaction and to imagine that conditions are known only to the individual agent. This is not strictly the case for every tag and condition; in fact it has already been violated by the defense tag, but this will merely be a helpful idea to keep in mind while discussing further levels of the model.

In this second level of Echo, when two agents on a site are paired for interaction, each compares its exchange condition to the offence tag of the other. If each condition is satisfied the interaction proceeds as in level 1. If neither condition is satisfied the trading interaction does not take place. If one condition is satisfied and the other is not, Holland suggests that the agent with the unsatisfied condition be given a chance to "flee" and that this chance could be simply a fixed probability, which becomes a global parameter of the simulation.
This conditional exchange interaction provides agent populations with a kind of evolvable interaction filter, allowing them to develop immunity to a population of predators, for example.

2-5 Level 3: Resource transformation

It can occur that a particular type of resource is in short supply within the Echo world, limiting agents' reproduction potential because of the extra time required to collect enough of this scarce resource for chromosomal replication. Another problem similar to this occurs when a particular population of agents requires significantly more units of a particular resource to copy their genome. This will also hinder reproductive potential for the same reason. One way for agents to adapt, given difficulties like those above, is to evolve a genome with resource amounts in a proportion that more closely matches that of the resource fountains in the surrounding environment. For example, if resource "b" is
extremely rare then those agents whose genomes contain fewer occurrences of “b” will have an advantage.

With this second extension to the basic Echo model, Holland provides his agents with another option. The new “gene” added to the chromosome in this level is intended to endow agents with a counterpart to chemical enzymes, giving them the ability to convert one resource type into another. As with all things in Echo, this enzyme gene does away with real world details such as metabolism and chemistry, stripping away the specifics of any given CAS in favor of capturing the essentials.

As enzymes are capable of catalyzing chemical reactions, so it is with this new enzyme gene. In an environment in which a particular resource is in short supply, agents may evolve an enzyme capable of converting a plentiful resource into a scarce one, thus speeding up or “catalyzing” agent reproduction. This new gene also opens up interesting possibility for later levels of the model in which hierarchical structures called multiagents and aggregates may evolve. In such cases, the enzymes may come to serve a role beyond simple compensation for resource shortages in the external environment.

This new gene is part of the control region of the agent chromosome. It must contain, at the very least, two letters, one to specify the resource that will be transformed and one to specify the product of the “reaction”. The cost associated with this transformation is that the agent must collect enough resources to create the enzyme string. It seems reasonable to impose a transformation cost on the model simply by specifying some
minimum length for enzyme genes. Holland suggests that the rate of transformation be sufficiently high to allow even short-lived agents to benefit.

Figure 5: Example of an enzyme gene

2-6 Level 4: Agent boundaries and adhesion

At this point, a new construct called a "boundary" is introduced into the mix. A boundary can be thought of as a wrapper or encapsulating membrane for agents. With the introduction of boundaries, and a new conditional adhesion interaction, Holland makes possible the joining together of groups of agents into transient multilevel entities called aggregates. This feature is intended to allow higher levels of organization in Echo beyond that of single agents and to provide a loose counterpart to such biological phenomena as colonial organisms and slime molds, in which groups of organisms unite to form a single entity to the benefit of each member of the aggregate.

Each agent is assigned to one specific boundary, though a boundary may hold many agents as well as other boundaries. This is more than simply a Russian doll or onion type of layering. Boundaries may contain any number of internal boundaries, forming a hierarchical tree-like structure. Aggregates migrate as a single unit and only those agents
assigned to the outer-most boundary of an aggregate may absorb resources from a site
and interact with other agents in the outer-most boundaries of other aggregates located on
the site.

A new tag called the "adhesion" tag is added to the tag region of the chromosome. It
is this tag that is the instrument of adhesion interactions, which lead to the formation of
aggregates containing multiple agents (lone agents, since they are still assigned to a
boundary of their own, can be thought of as aggregates containing only a single agent).
Holland recommends that this interaction take place between mother and daughter agents
at the time of asexual reproduction. In this way, it is hoped, aggregates may grow from a
single "seed". The possibility of adhesion interactions between strangers, is given as an
optional variation on the basic model.

Figure 6: Interaction in level 4

The adhesion tags of the mother and daughter are paired with the offence tags of the
other. The outcome of the interaction is one of three possible boundary configurations,
determined as follows:
• If both match scores are low, there is no adhesion, and the daughter agent is ejected from the aggregate in which the mother resides.

![Diagram of daughter agent being ejected from aggregate](image1)

**Figure 7: Daughter agent is ejected from aggregate**

• If both match scores are high, the daughter agent is assigned to its mother’s boundary.

![Diagram of daughter agent sharing boundary with mother](image2)

**Figure 8: Daughter agent shares boundary of mother**

• Finally, if one match score is high and the other is low, the agent with the low score is engulfed by the other. The engulfed agent is given its own boundary, which becomes an internal boundary of the engulfing agent’s boundary.

![Diagram of daughter agent being engulfed by mother](image3)

**Figure 9: Daughter agent is engulfed by mother**
If the adhesion tag of one agent were simply paired with the adhesion tag of the other then match scores would always be high. It would then follow that agent reproduction would consistently result in the formation of aggregates with the second type of boundary configuration depicted above. It is for this reason that the adhesion tag is matched against an offence tag, allowing all types of adhesion, including none at all, to occur.

A convenient way to denote and depict the structure of aggregates is to use a tree diagram where each node represents a single boundary, each branch an internal boundary and each node is labeled with the number of agents assigned to that boundary. The root of such a tree would correspond to the outer-most boundary of the aggregate. An example of this is shown below.

![Tree Diagram](image)

**Figure 10: An aggregate and its corresponding tree representation**

A ramification of introducing aggregation into the model is that pairing agents for interaction becomes slightly more complicated. A new concept referred to as the "domain of interaction" is introduced as an aid in dealing with this complication. When it comes time to choose an interaction partner for an agent, the partner is chosen randomly from
within the agent’s domain of interaction. The domain of interaction consists of the following groups of agents:

- Agents assigned to the same boundary as the agent
- Agents assigned to the parent boundary (consider the tree diagram representation of aggregate structure described above)
- Agents assigned to boundaries which are immediately interior to the agent’s own boundary
- Agents assigned to boundaries that are adjacent. For agents in the outer-most boundary of an aggregate, adjacent boundaries are the outer-most boundaries of all other aggregates on the same site. For all other agents, the adjacent boundaries are those sharing a common parent boundary with agent’s own.

![Figure 11: An agent's domain of interaction](image-url)
As was mentioned previously, aggregates migrate from site to site as a single unit. Determining when an aggregate should migrate, however, is an issue not touched on by Holland is “Hidden Order”.

2-7 Level 5: Selective mating

At this point, the only genetic variation arising in Echo, beyond that of the initial few genomes directly introduced at the outset of executing a given implementation of the model, stems entirely from random mutation occurring during agent reproduction. This fifth level adds genetic crossover as a second source genomic variety.

As is the case with all five extensions to the basic Echo model, this one brings with it a new addition to the chromosome. This time the addition is a new condition in the control region called the “mating condition”, which operates in the same manner as the exchange condition. Like the exchange condition, the mating condition is also paired with the offence tag of the partner agent. Both conditions must be satisfied for mating to proceed.

![Diagram of Agent Interaction in Level 5]

**Figure 12: Interaction in level 5**
There are many ways to determine when mating should take place. One way, suggested by Holland, is when an agent has collected enough resources to duplicate its chromosome. It attempts to locate a mate within its domain of interaction, one that is also ready to reproduce. Two offspring are then generated and crossover is performed on the chromosomes, followed by mutation. Another way to accomplish this might be to break the connection between mating and reproduction, allowing genetic crossover without producing offspring.

2-8 Level 6: Conditional replication and multiagents

It is this final extension of Echo that introduces a simple model of the process of metazoan embryogenesis, the development of a multicellular organism from a single cell. Naturally, the deliberate discarding of the details of real world phenomena, and the attempt to cut away all but the essentials, is particularly evident in this level. It is not just biological evolution that has prompted this extension. All CAS exhibit, in some form or another, increasing organization beyond their lowest levels as they evolve. Nevertheless, it is clear, from both the structure of the model framework and the terminology used, that metazoan embryogenesis is the primary CAS phenomenon that most readily maps to the specifics of this level, and that has guided the design to the greatest degree.
A new kind of mutation is introduced into the model, allowing agent aggregates to become “multiagents” via concatenation of their chromosomes. A multiagent differs from a standard aggregate in several important ways:

- They have a shared reservoir of resources
- They reproduce as a whole
- They possess a genome consisting of multiple agent chromosomes

The agents within a multiagent are referred to as “agent compartments” or just “compartments”. The genome of a multiagent is a concatenation of agent chromosomes, but with the addition of three new genes for each chromosome. The first is a new condition called the “replication condition”, the second is a single letter called the “active marker” and the third is another single letter called the “satisfied marker”.

When a multiagent reproduces, only those compartments, in the genome will be “expressed”, whose replication conditions are satisfied by the offence tag of some currently present agent compartment in the mother. For example, if a multiagent’s genome consists of three agent chromosomes, “c1”, “c2” and “c3”, but “c2”’s replication condition is the only one satisfied by the offence tag of some agent compartment present in the multiagent, then the offspring, though still possessing an identical genome, will only contain agent compartment “c2”. This provides a simple model of gene repression and derepression. Going further with this analogy from biology, one can think of replication conditions as chemical “switches”, waiting for the proper molecule to come
along, thus activating a particular gene. When a gene is active, or derepressed, the chemical machinery of the cell translates it into whatever substances it encodes. The offence tags of the active compartments in the mother multiagent are like the molecules that bind to the repressed genes' switches, derepressing them and causing them to be expressed in the daughter multiagent. It is then possible for this daughter multiagent to reproduce and for the granddaughter to have a different set of agent compartments present, and so on. This is similar to the variety of differentiated cell types produced in multicellular organisms (for example, brain, muscle, skin, fat... etc), which, nevertheless, have the same genetic material in their nucleus.

The addition of the "active" and "satisfied" markers, for each chromosome, is done merely to facilitate the process described above. The "active" marker will be set to "true" ("true" and "false" are simply designated by two different resource letters) for each chromosome in the genome for which there is an agent compartment present in the mother multiagent. The "satisfied" marker will be set for every chromosome in the genome for which the replication condition has been satisfied. These markers are then used to determine the structure of the daughter multiagent.
Figure 13: Illustration of multiagent reproduction

One of the central concerns of the Echo model is the question of how complex hierarchical structures evolve. In Echo, aggregation acts as a testing ground in which individual “building blocks”, the agents, experiment, via the filtered trial and error of evolution, with various relationship configurations. Mutations producing multiagents from aggregates act as a kind of glue, creating further building blocks of greater complexity. These, in turn, continue experimenting with further configurations. This process is related, conceptually, to the evolution of modules in genetic programming.
(GP). Selection pressure in the system eliminates poorer modules, or building blocks, and favors the development of more “useful” ones. In GP, this builds programs; in Echo it builds complex hierarchical multiagents.

2-9 Previous Work

Part of the motivation for this project stems from past study done using the Echo model. Both [Bedau 98] and [Hraber] give their own implementations of Echo, each of which is very minimalist with regard to the subset of the six levels implemented. Both implementations possess levels 1, 2 and 5 only. That is, offence, defense, conditional exchange and selective mating. Resource transformation, aggregation and multicellular components of the model are absent in both cases. An interesting aspect of both experiments involves a “shadow” or “neutral” model, which runs alongside the real model. These shadow models reflect the activities of the real models except that selection pressure is random with respect to genomes, allowing a comparison, which emphasizes the role of the evolutionary component of the models in the behavior they exhibit.

The Echo model in [Bedau 98] consists of only a single site in which all agents reside and in which all interactions take place. Agents reproduce as soon as possible, that is, as soon as enough resources have been collected to duplicate the chromosome. There is a fixed percentage of the leftover surplus after reproduction, which is transferred to the daughter agent. Those resources that an agent can absorb from the site are determined by a special gene called the “uptake mask”, which is comprised of a string of bits instead of
resources letters. There is a single bit for each type of resource, with 1’s indicating an ability to absorb that resource. Agent interactions are divided into combat, mating and trade. A single-letter gene called the “trading resource” determines the resource exchanged during trade. Single-point crossover is the chosen method for successful mating interactions.

In [Bedau 98] the data from the model is analyzed with the application of a measure of long-term evolutionary dynamics. They had applied similar measures to their own model, called “Strategic Bugs” [Mitchell], [Bedau 92]. Their technique, which involves measures of diversity and evolutionary activity, groups the dynamics into three classes: no adaptive activity, bounded adaptive activity, and unbounded adaptive activity. Their analysis, which also involved calculations regarding the Phanerozoic fossil record, for comparison, concluded that Echo displays bounded evolutionary dynamics, in contrast to unbounded dynamics as seen in the fossil data.

The implementation described in [Hraber] was very similar to that described above. It included a bit-string uptake mask, the division of agent interactions into combat, mating and trade, and a single-letter trade resource gene. Some points of difference include a fixed “self-replication threshold” allowing agents to accumulate a larger surplus of resources before replicating (see the discussion of replication factors in section 3-6), two-point instead of single-point crossover and a measure of proximity between agents on a site, in which the probability of interaction between a pair falls off with increasing distance.
The data from runs of this second model were also compared to similar data from the field of biology. This time, instead of measuring evolutionary dynamics, the behavior of Echo was compared to natural systems using a measure of species abundance distribution, called the Preston curve, a canonical distribution, which essentially reflects the common observation that most species sampled will be rare, whereas a small few will be quite common. Another comparison was also made using the species area-scaling relation, which reflects distribution changes as the sample area increases. In both cases Echo exhibited output that was qualitatively in agreement with natural data.

In this project, the complete Echo implementation will be tested in another way. In order to establish whether or not the model exhibits CAS-like behavior, the output data will be analyzed to see if it can be shown to be exhibiting any familiar CAS phenomena.
3. An Echo Implementation

This section describes the specifics of the Echo model, which I have implemented. The primary focus will be on the following aspects of the model:

- Those which differ from Holland's description of the Echo model and the reasons for those differences

- Those that were ambiguous or unspecified in "Hidden Order"

- Issues and design decisions which arise when making a concrete instance of an Echo model and the reasons for the choices made

Sections 3-1 through 3-4 discuss the 4 classes created to implement the model, "World", "Site", "Agent" and "Boundary", respectively. Section 3-5 deals with the data structure used to track genomes, populations and ancestry, the "STTree" class. Finally, sections 3-6 through 3-11 describe the implementation of each of the six levels of the model (see Sec 1-3).

The programming was done using Borland's C++ Builder software (copyright 1997, Borland International, Inc) running under Windows ME on a 600Mhz PC with 256MB of RAM. Unless otherwise specified, all testing of the system was done using the same set of default global parameters, which I have chosen because of the subjectively interesting results they produce. These tests may not reflect the full range of potential behavior of the system.
3-1 The Echo World class

There are a variety of global parameters in this implementation that are stored in a class called "World". Some of which come from Holland directly, such as the rate of random agent deaths or the number of resources types, while others are the result of various design decisions that went into shaping this particular version of Echo. These parameters can be divided into two categories: dimension-like parameters and probability parameters. Dimension-like parameters are those, which in some way stipulate the size of the Echo world, and are non-negative integer values whereas the probability parameters affect the rates of a number of events in the simulation, and are floating point values in the interval [0,1], with 0 indicating that the event will never occur and 1 indicating the event will occur at every opportunity.

3-1-1 Dimension-like parameters

1) Number of rows and columns

The chosen layout for the world in this model is a simple two-dimensional array of sites, each connected in the north, south, east and west directions to its immediate neighbor sites. These parameters determine the size of this array, which is toroidal when it comes to agent migration. This layout was chosen because it seemed to be the most obvious and straightforward layout. Each site is connected to exactly four other sites. This simplifies the coding, in comparison to other possible layouts, which may have an arbitrary number of links between sites. Note that in several previous implementations of Echo found in the
literature, such as in [Hraber] and [Bedau 98], the implemented model has been restricted a world consisting of only a single site.

2) **Number of resources**

This parameter specifies the size of the resource alphabet. In most cases, a value between four and seven seem to produce “pleasing” results. Less than four seems too simplistic and greater than seven seems to seriously hinder the survival chances of those initial few randomly generated agents placed into the world at the beginning of a run. The graph on the right shows the results of executing the simulation ten times for resource alphabet sizes between four and eight, with the world being seeded by a population of 40 randomly generated agents, and then counting the number of “stagnant” runs, defined as runs in which the total world population never exceeds 100 within the first 1000 time cycles. Such a run indicates that the initial 40 agents are unable to effectively reproduce, and are therefore unable to evolve. These runs used a 20x20 world and a tag length (see below) of five. This
stagnation problem for large resource alphabets can be overcome with larger initial populations, which increase the chances of finding a good "seed", or through seeding the run with pre-designed agents whose chromosomes are not randomly generated. The manner in which agents absorb resources from sites in this particular implementation (see section 3-6) is primarily to blame for this issue.

3) Tag and condition length

I have chosen to enforce a fixed tag length as a global parameter for the sake of simplicity. This frees the model from issues such as keeping track of gene offsets within an individual agent's chromosome (these offsets are now universal and need only be calculated once) and from the added complication of performing crossover on chromosomes of different lengths. This parameter applies to tags and conditions only. Values that seem to produce the most interesting dynamics are between three and eight.

4) Maximum site occupancy

This parameter specifies the maximum number of agents, which may be present simultaneously on a site. When one executes the model for the first time, there are several parameters one may wish to adjust in order to strike a balance and induce desirable evolutionary dynamics in the system. One example of this is the probability of mutation during reproduction. If this probability is set too low, one may find that the system evolves too slowly. Setting it too high may prevent any
one population from gaining a foothold because it is being swamped by numerous small populations of mutant agents. Maximum site occupancy, though not a necessary parameter for the system, is like an extra tuning knob at the user’s disposal, placing a soft limit on the total number of agents that may be present in the world. Other ways to tune the system, with regard to world population size, are via the resource production rates on each individual site, the probability of taxation (see sections 3-1-2 and 3-2) and the probability of random agent death (see below).

3-1-2 Probability parameters

1) Taxation probability

Agents in this model are subject to a kind of “metabolic” taxation. This tax takes the form of a single unit of a particular resource type. Taxation is probabilistic and occurs for each agent at each time cycle with a probability dictated by this parameter. Should an agent be unable to pay a requested tax because it does not possess a unit of the requested resource in its reservoir, it is removed from the population and its resources, both genetic and reservoir, are transferred to the site for absorption by other agents. This is one way to prevent “freeloaders” in the Echo ecology. A freeloader is an agent, which does not reproduce, but simply sits at a site and absorbs resources. Taxation may not effectively kill all freeloaders since agents may exist, which absorb the taxed resource of their site, but which are unable to absorb other resources necessary for reproduction.
2) Random agent death probability

Random agent death occurs in a manner similar to taxation. It applies to each agent, or agent compartment in a multiagent, at each time cycle with a probability dictated by this parameter. A killed agent is removed from the population and its resources are recycled in the manner described above. This parameter serves two purposes. The first is to ensure that freeloaders are eventually eliminated. The second is to act as a tuning knob, allowing the user to strike a balance between death rates and birth rates. Though Echo is intended to be a model with minimal tweaking to produce desired behaviors, some degree of this is inevitable.

3) Mutation probability 1

There are three distinct types of mutation in this model. The mutations controlled by this parameter are mutations at the time of agent and multiagent replication. In the case of agents this is restricted to point mutations since agent chromosome length is fixed. For multiagents this can result in a point mutation, a duplication of some compartment in the genome, or the deletion of some compartment. For simplicity, deletion mutations in multiagents may only occur for compartments that are not active (see discussion of the active marker in section 2-8). In the event of a compartment duplication mutation, all agent compartments assigned to that part of the genome are reassigned to the duplicate compartment (see discussion of compartment number in section 3-3) with probability 0.5.
4) Mutation probability 2

This second type of mutation, which can occur at any time, is intended to help counter the problem of freeloaders and to reduce the chance of prolonged stagnation in the world. If the agent population should happen to settle into a group of freeloaders, which happens on rare occasions, this mutation is meant to give evolution a chance to kick-start itself once more by hopefully transforming a freeloader into an agent capable of self-reproduction. Without such a mutation, a population composed only of freeloaders is doomed to eventual extinction or indefinite stagnation.

5) Mutation probability 3

This third type of mutation converts an aggregate into a multiagent. When such a mutation occurs, the genetic information of every agent and multiagent within the aggregate is concatenated into a single genome. The newly formed multiagent is given a common reservoir, which is shared by all agents, now referred to as agent compartments, within its outer-most boundary and all internal boundaries.

6) Interaction probability

This probability governs the number of agent pairings for interaction that will occur in a given time cycle. In each cycle of the simulation, the system traverses each agent, aggregate and multiagent present on a site, and each agent or compartment within an aggregate or multiagent, and uses this parameter to determine if a partner will be chosen from within the agent's domain of
interaction. Thus, with an interaction probability of 0.5 and 20 agents on a given site, one can expect about 10 pairings to occur. Note that, on average, the number of pairings will be slightly less than 50% of the number of agents present since some pairings will result in a fatal combat interaction, reducing the number of agents present.

7) Migration probability

Migration probability governs the chance that a hungry agent, aggregate or multiagent will migrate from the site where it is located. A hungry agent is one that was unable to absorb any resources during the last time cycle. A hungry aggregate is not so straightforward and has been defined, in this model, as an aggregate for which at least three fourths of the agents in the outer-most boundary are hungry. The same definition of hungry is used in the case of multiagents as well. This migration parameter was introduced to slow the "geographic" spread of a growing population. Another way to view this parameter is as measure of agent mobility, with the value 1.0 indicating agents that never hesitate to migrate when they are hungry, and with lesser values indicating agents that are more reluctant to abandon a site.

8) Adhesion interaction probability

In this model, agent adhesion interactions occur between mothers and daughters at the time of reproduction and also between neighbors during normal interactions. The reason for including this second type of adhesion interaction is discussed in
section 3-9. A value of 1.0, for this parameter, indicates that paired agents will always be checked for adhesion, and 0.0 indicates that this type of adhesion will never occur.

There are other instance variables for the World class, which are not simulation parameters. These can be divided into world statistics and gene offsets, described below.

3-1-3 World statistics

1) Total world population

This variable is constantly updated to reflect the current total world population. Lone agents and agents within an aggregate are counted, but compartments within a multiagent are not, the whole multiagent is counted as a single entity.

2) Number of multiagents

This is similar to total world population, but gives the total multiagent population only.

3) Number of boundaries

This variable gives the precise number of Boundary objects (see section 3-4) in the world. For a population consisting entirely of lone agents, this number will equal the total world population. When this value becomes less than the total world population, this is an indication that there is some agent aggregation
occurring. Should this value ever exceed the total world population, which can occur but only extremely rarely, this indicates that a large amount of the ecological "biomass" is stored within multiagents who have a structure that includes internal boundaries. Thus, comparing this value to the total world population gives some hint about the relative proportion of aggregates, agents and multiagents.

4) *Number of internal boundaries*

This is similar to the number of boundaries, but counts only those boundaries, which are internal to other boundaries. This gives the user a sense of the amount of layering occurring within aggregates and multiagents. Preliminary testing of the system typically results in this value being a very small percentage of the total number of boundaries. Executing the simulation for 20 non-stagnant runs, of 1000 cycles each, gave an average of only slightly more than 1% of all boundaries being internal. Possible reasons why complex layered structures are so rare are discussed briefly in section 3-9.

5) *Number of agents sharing boundaries*

This value tracks the number of agents that are sharing a boundary with other agents. This is another informative variable, which gives the user an idea of the degree of aggregation occurring in the simulation.
6) *Combat, trading and mating totals*

These three variables keep a running total on the number of combat, trading and mating interactions that have occurred since the beginning of the simulation.

7) *Death totals (combat, escapes, taxation, random)*

These four totals cumulatively track the total number of deaths due to combat, instances of agents escaping from combat, deaths due to taxation and random deaths, respectively.

3-1-4 Gene offsets and other related variables

Since tag and condition lengths are parameters of the system, but remain fixed throughout a given run, the offsets of each gene in an agent chromosome are calculated before the run begins. These offsets are stored in the World object and never need to be recalculated. The offsets of the three extra genes, which are added to agent chromosomes when they concatenate to form a multiagent genome, are also stored in the World object together with the lengths of both chromosomes and compartments. Thus, in order to access, for example, the \( n^{th} \) letter of the defense tag on the \( i^{th} \) compartment in a multiagent genome, its offset is calculated as: \(((i-1) \times \text{compartment length}) + \text{offset of defense tag} + (n-1)\)
3-1-5 Logging

As the simulation proceeds, a population log file is continuously written. At the end of each time cycle, the population of each “living” genome in the population is written to the file. By renaming this log file and loading it into a custom graphing utility in the program, the user is able to produce a population graph of the run, with populations either overlapping or stacked. The following two figures show both types of graphs, generated from the log file of a 2000-cycle run.

Figure 15: Population graph of a 2000-cycle run
Figure 16: Stacked population graph from a 2000-cycle run

The population log file is generated using a genome population tracking data structure. This data structure tracks the population of every genome that has occurred during a run, among other things, such as ancestry information. The genome tracker is discussed in more detail in section 3-5.

3-2 The Site class

Site objects come in three types. The first two types are concerned with the way agents are taxed. A site may charge a normal or a random tax. A site with normal taxation always charges the same tax, a resource letter specified by the user before the simulation begins. A site with random taxation charges each resource type with equal probability. The third type of site is called a “barrier” site. A barrier site cannot contain agents. By placing barrier sites in world, the user can create more restrictive geographies. The figure
below shows a 20x20 world in which barrier sites (depicted by small gray squares) have been used to increase the maximum migratory distance between two sites in the world to 100 instead of 19, which is the maximum on a 20x20 toroidal lattice without barriers. The colouring scheme used below for non-barrier sites is explained in section 3-3.

![Image](image_url)

**Figure 17: Use of barrier sites to alter world geography**

Each Site object holds four arrays that control the storage and production of resources. The first array stores the initial amount of each resource type that is present when the simulation begins. The second holds the current amount of each resource type. A third stores the production rates for each type, and a fourth determines the maximum amount of each type that may accumulate on the site. These four arrays are intended to provide the user with a high degree of control over resource production and accumulation on the site. The second and third arrays store floating-point values, as opposed to
integers, in order to provide greater control over resource production rates. Resources may be produced on a site in fractional amounts, and thus at rates that are less than one unit per cycle. Agents will only absorb resources in whole number amounts. See section 3-6 for a discussion on resource absorption by agents.

Site objects maintain an array of boundaries. These are the agents, aggregates and multiagents that are located at the site. This array has a fixed size, which is the same for all sites in the world. Agents are not permitted to migrate to, or be born in sites that are full. When an agent reproduces, the daughter may be placed in the same site as its mother or in a site immediately connected to it. If all of these sites are either full or barriers, the offspring is considered to have been “crowded” to death, and its reservoir and chromosome resources are transferred to the site.

Site objects track three interaction statistics. These three values are the number of combat, trade and mating interactions that have occurred in the last time cycle. This information is used to provide the user with a variety of views on the Echo world. The figure below shows a snapshot of a 20x20 world where the brightness of each site is based on the number of trade interactions that have occurred there in the last time cycle.
3-3 The Agent class

The Agent objects in this implementation possess more than just a chromosome, a reservoir and a pointer to their assigned boundary. There is a Boolean flag indicating whether or not the agent is hungry. At the beginning of each time cycle this flag is set to true and becomes false when the agent obtains resources, either through absorption, trade or combat. Another Boolean flag, possessed by each agent, indicates whether or not the agent is an agent compartment in a multiagent, and must therefore make appropriate use of its pointer to the shared reservoir of its multiagent. Agents also contain a pointer to a single leaf node in the World object’s genome tracking data structure. This leaf node is responsible for maintaining statistics about the agent’s genome, such as the time of its
first appearance, the time it becomes extinct, its ancestor genome(s) and so forth. A
discussion of the genome tracking data structure can be found in section 3-5. When an
agent is born, it is assigned the tracker node corresponding to its chromosome. When it
eventually dies, it uses this pointer to inform the tracker node, which then updates
statistics for that agent’s genome.

For the sake of providing an interesting visualization in the user interface, each agent
is assigned one of three possible colours, based on its chromosome. The agent’s
chromosome string is interpreted as an array of integer values and these are summed
together. This sum is divided by three and the remainder determines the agent’s colour;
zero for red, one for green and two for blue. Since this colour is the same for all agents
with the same chromosome, the value is stored in the agent’s genome tracker node. The
user interface does not permit the viewing of individual agents, but rather, as the
simulation proceeds, the user sees a colour for each site, which is a combination of the
colours of the agents located at the site. For example, if there are an equal number of red
and blue agents on a site, but no green agents, the site is displayed as purple, an equal
combination of red and blue. The user is provided with three colouring scheme
variations. The first, called “raw”, is a straightforward combination of red, green and blue
in proportion to the number of agents on the site that are assigned those colours. The
brightness of the colour is based on the total number of agents of all colours, relative to
the maximum site occupancy value. Empty sites, therefore, have zero brightness and are
coloured black. The second colouring scheme, called “enhanced” is similar to “raw”,
except that the dominant colour on the site is given greater representation. For example, if
a site contains 10 red agents, 10 green agents and 11 blue agents, the "raw" colouring scheme shows a very dull gray with a hint of blue, whereas the "enhanced" scheme is clearly blue. The third colouring scheme, called "majority" simply displays the dominant colour only and at full brightness.

Agent objects possess two instance variables that are used exclusively by agent compartments. The first is the agent's compartment number. This simply indicates which chromosome in the multiagent’s genome dictates the compartment’s behavior. The second is a pointer to the multiagent’s shared reservoir. Agent compartments make use of both the shared reservoir of their multiagent and their own “personal” reservoirs. The reason for this is that, while it is desirable to allow the multiagent to reproduce as a whole, it is also desirable to allow an individual multiagent to grow. In order to accomplish this, the resources in a multiagent’s shared reservoir are divided into those, which are “claimed” and those, which are “surplus”. For each unit of resource in a compartment’s “personal” reservoir, there is a corresponding unit of resource in the multiagent’s reservoir, and this unit is “claimed” by that compartment. The surplus resources in a multiagent’s reservoir accumulate whenever a compartment fails to pay tax or is randomly killed. Its resources, instead of being transferred to the site, are transferred to the shared reservoir. Since these resources are not claimed, they may not be used by individual compartments for replication, but only by the entire multiagent. In this way, individual compartments may replicate and allow the multiagent to grow, while at the same time, the multiagent will eventually be able reproduce as a whole. The replication of individual compartments helps to prevent them from disappearing in the multiagent,
which is important, as they may be vital to its survival. On the other hand, a large number of compartments can become a resource drain on the multiagent. Thus, it is hoped, through taxation and random death, the growing population of components within the multiagent is kept in check while simultaneously increasing the amount of surplus resources used in replicating the whole. This also means that improved compartment reproduction leads to improved reproduction of the entire multiagent. This situation is illustrated below.

![Diagram of multiagent and reservoirs](image)

**Figure 19: Claimed vs. Surplus resources in a multiagent**
3-4 The Boundary class

Boundary objects are containers holding both an array of agents and an array of internal boundaries. They are, however, more than simply container objects. Once a mutation converts an aggregate into a multiagent, suddenly the mutated boundary object must take on a role similar to that of an Agent object. It will have its own genome, its own genome tracking node and its own reservoir, which is shared among all of its compartments, as was described in the previous section.

In order to more easily access members of an agent’s domain of interaction within an aggregate or multiagent, each boundary is given a pointer to its parent boundary (see the discussion on using tree diagrams and terminology to describe aggregates in section 2-6). There is also a root pointer, which is used by boundaries within a multiagent to quickly access its outer-most boundary, which is important since it contains the multiagent genome, tracker node and reservoir.

Aside from pointers to a genome, tracker node and shared reservoir, the outer-most boundary of a multiagent possesses two more noteworthy instance variables. The first is an integer indicating the total number of agent chromosomes, which have been concatenated into the genome. The second is also an integer, which holds the replication factor for the multiagent. The replication factor is used to determine when the multiagent possesses enough resources to reproduce. A value of one indicates that the multiagent need only collect enough resources to duplicate its genome, at which point it will reproduce. A value of two indicates that the multiagent must collect twice the amount
needed to duplicate its genome before it will reproduce. A value of three indicates that thrice the amount is needed, and so forth. The value of the replication factor is calculated as one less than the average of the replication factors for each compartment in its genome, with the restriction that it may not be less than one. Subtracting one from the average is a way of “suggesting” that reproduction of the whole multiagent is more important than reproduction of its individual compartments. See section 3-6 for a discussion of the replication factors of individual agents.

3-5 Genome tracking

Much of the information about a given run of the simulation is managed by a data structure called the genome tracker. The tracker is a kind of augmented trie tree with path compression. For each distinct genome that has appeared in the simulation, there is a node in the tracker, which maintains the following information about that genome:

- Its current population
- Its colour
- A linked list of “origin” nodes

Every agent and multiagent is given a pointer to its corresponding tracker node. It uses this pointer to inform the tracker of its death. The tracker node then subtracts one from the population. If the population has become zero, an extinction has occurred and this information is recorded in an origin node.
An “origin” node holds information about the appearance and extinction of a particular genome. It is possible for a genome to appear and go extinct several times over the course of a run. For this reason, origin nodes are chained together in a linked list, with each node describing a single appearance and disappearance of the genome in the population. Each origin node holds the following information:

- The number of immediate ancestors of the genome
- The time of its appearance
- The time of its extinction
- Pointer(s) to the tracker nodes of the immediate ancestor genome(s)
- The type of origin

The “type” of an origin node depends on the way in which the genome came into existence. This can be any of the following:

- **Original**: A genome used to seed the initial population
- **Mutation**: A genome that resulted from the mutation of a single ancestor
- **Crossover**: A genome that resulted from crossover between two others
- **Burst**: A genome that appeared when a multiagent’s outer-most boundary “burst” (see section 3-9 for an explanation of “bursting”). When this occurs, the agent compartments within the multiagent become normal agents once again. Those that are not immediately “crowded” to death (see section 3-2) are
tracked as new genomes. This type of damage could evolve to become beneficial for the multiagent population as whole. Should the new agents, resulting from a burst multiagent, become predators of those agents responsible for the burst in the first place, this may induce the evolution of a relationship similar to certain situations in biology where a species protects itself from predation simply by being toxic. The few members of the species that are unfortunate enough to be devoured are so toxic to the predator, that this becomes a selective pressure against predation in the first place.

- **Eject**: An agent compartment within a multiagent has reproduced and undergone an adhesion interaction with its daughter compartment. The daughter compartment has been ejected from the multiagent and exists now as a regular agent. In this way, it is possible for a multiagent to become a kind of "factory" producing agents of another type. Should these ejected agents be suitable prey for another population, the potential exists, then, for the evolution of a form of trade between multiagents, using entire agents instead of resources. This type of agent-ejecting could also lead an arms race, in which a multiagent uses ejected agents to fight a competing "opponent", for example, this opponent may, in response, evolve a counter-attack, and so on.

With this information, it is possible to construct a "family tree" and an accurate chronology for the evolution of the genomes in the system, tracing each genome all the way to a subset of the original seed genomes.
The genome tracker also maintains some extra global information in its root node. This includes the number of distinct genomes present in the population, the number of extinct genomes, the top ten genomes, ranked by population, and also information about the size of the trie tree itself.

A trie tree was chosen as the data structure to manage genome tracking, because agent chromosomes have a fixed length, making insertion and search operations run in constant time. Put another way, search and insertion operations run in linear time with respect to the length of agent chromosomes, but this is constant for any given run. The only exception to this occurs with multiagents, which may have genomes many times larger than a single agent chromosome. Multiagents, however, are very much a minority in Echo. Single agents tend to dominate the population. Possible reasons for why this may be are discussed in section 3-9.

![Trie Tree Diagram]

*Figure 20: Example of a trie tree (29 nodes)*

Despite the rarity of multiagents, the genome-tracking tree has been optimized with path compression. For particularly large genomes, and for large numbers of genomes, this
can reduce the required memory significantly. Path compression in a trie tree involves compressing any node chains into a single node. The figure below shows a compressed version of the trie tree from the previous figure. This optimization is important since Echo can easily generate tens of thousands of genomes in a very short time.

![Trie Tree Diagram]

**Figure 21: Example of a trie tree with path compression (9 nodes)**

### 3-6 Offence, defense and resource absorption

One aspect of this model that differs from Holland’s description in “Hidden Order” is the way in which tags and conditions are used to determine the outcome of agent interactions. In Holland’s model, tags and conditions are composed of different subsets of the resource alphabet. Conditions are either satisfied by the tag they are matched against or they are not satisfied. In this model, however, I have decided to go with a different approach. In this implementation, both tags and conditions, and all other genes for that matter, are composed of strings of resource letters taken from the entire resource alphabet. A benefit of this approach is that one need not be concerned about which gene a point mutation is affecting. Any letter in an agent chromosome may be replaced with any other and the result is still a valid agent. Thus, there is no difference between a tag and a
condition in this model other than the conceptual difference whereby a tag is considered to be "visible" to other agents and a condition is not. There are also no wild card resources. Conditions are not simply satisfied or unsatisfied, but satisfied to a particular degree, which is determined using the hamming distance between the two matched strings. Interaction outcomes are always probabilistic, with probabilities that increase as hamming distance decreases. It is hoped that this probabilistic approach to agent interactions will provide a gradient, which evolution may use to allow agents to "home in" on particular genetic solutions. For example, if an agent population requires three mutations in its exchange condition in order to effectively avoid interactions with a predator population, each step along the way will provide a survival advantage.

Another variation on Holland's model is the inclusion of a "replication factor" in the agent chromosomes. The replication factor determines how many resources are required before an agent will reproduce. The replication factor "gene" consists of only two letters from the resource alphabet, which are treated as zeros and ones, depending on whether their integer values are even or odd. They are then combined to create a binary number between 0 and 3, which is increased by one, making the range 1 through 4. This number is the replication factor for the agent. Let "n" be the size of the resource alphabet, let \([r_1, r_2, r_3, \ldots, r_n]\) be the amount of each resource type needed to duplicate the agent's chromosome, and let "f" be the agent's replication factor. Then \([fr_1, fr_2, fr_3, \ldots, fr_n]\) is the amount of each resource the agent must collect before it will reproduce. This puts the replication factor in the hands of evolution, instead of making it a fixed parameter of the system. As a preliminary test, the system was run for 20 non-stagnant runs, and the
replication factor of the leading population after 1000 cycles was recorded. The figure below shows the results of this test, indicating that a high replication factor tends to be favored, and that, in this model at least, faster replication does not necessarily equate to evolutionary success.

Figure 22: Frequencies of replication factor values

In [Hraber] and [Bedau 98] the agents are given a bit-string called an “uptake mask”, which determines whether or not the agent may absorb each type of resource. An agent may absorb resource $i$ provided that there is a “1” at position $i$ in the uptake mask. I have built upon this idea to create an uptake “gene”, which is composed of resource letters, like the rest of the chromosome, instead of bits. Given the resource alphabet $\{a, b, c, d, e\}$, consider the two uptake genes in the figure below.

Figure 23: Two uptake genes
The agent with the first uptake gene can absorb all resources equally effectively, whereas the agent with the second uptake gene can absorb only “a”, “b” and “c”. However, the second agent can absorb resource “a” three times better than the first agent. The reason for this is that the resources on a site are distributed to each agent in a “roulette wheel” fashion, depicted below.

![Diagram of a roulette wheel with letters representing resources distributed to agents]

**Figure 24: Roulette wheel for resource distribution**

Thus, the probability that a given unit of resource will go to a given agent is proportional to the number of occurrences of that resource letter in the agent’s uptake gene. It is now possible for agents to specialize in absorbing a particular resource, at the expense of being forced to obtain other resources through trade or combat.

In [Hraber] agents are given a “gene” called the “trading resource”, which is a single-letter gene in the agent chromosome. The agent will trade only this resource type and no other. I have combined this idea with that of the “uptake gene”, described above, to create a “trading gene”. The trading gene works in much the same manner as the uptake gene. When an agent engages in a trading interaction, the resource type that it chooses to trade is chosen randomly from the letters in the trading gene. Thus, agents may evolve to trade
all resources, or may specialize, and trade only a subset of all possible resources. As with the uptake gene, it is left to evolution to determine which strategy to use. The simulation was executed for 10 non-stagnant runs. At the end of the 1000th cycle, the uptake and trading genes were recorded for the leading population. The figure below shows the results and gives an idea of the kind of variety that typically evolves.

<table>
<thead>
<tr>
<th>Uptake</th>
<th>Trading</th>
</tr>
</thead>
<tbody>
<tr>
<td>baedc</td>
<td>acbac</td>
</tr>
<tr>
<td>aaccc</td>
<td>eeaca</td>
</tr>
<tr>
<td>cbcde</td>
<td>cdbcb</td>
</tr>
<tr>
<td>addea</td>
<td>eaaea</td>
</tr>
<tr>
<td>cbaad</td>
<td>eddda</td>
</tr>
<tr>
<td>adbce</td>
<td>bbbdb</td>
</tr>
<tr>
<td>cabeled</td>
<td>eceba</td>
</tr>
<tr>
<td>eddae</td>
<td>ddbeb</td>
</tr>
<tr>
<td>ecade</td>
<td>eecbb</td>
</tr>
<tr>
<td>aecdb</td>
<td>acbdc</td>
</tr>
</tbody>
</table>

Figure 25: 10 examples of uptake and trading genes

3.7 Conditional exchange

The figure below depicts the breakdown of agent interactions into combat, trading, mating and adhesion. In [Holland] the combat and trading interactions are one and the same. In [Hraber] the breakdown is as show below, but without adhesion. The addition of the adhesion interaction reflects the choice to allow neighbors to adhere to one-another, not just mother and daughter agents.
The reasons for allowing neighbors to aggregate together, as well as mothers and daughters, are described in section 3.9. The choice to check first for agent combat before proceeding to other interactions is meant to reflect a more realistic organism-like behavior.
3-8 The enzyme gene

Preliminary tests during implementation of the model showed that most agents would evolve disabled enzyme genes; that is, enzymes that convert a unit of resource into a unit of the same type, effectively doing nothing. It seemed, therefore, that the enzymes were more detrimental than beneficial. To get around this problem, I have added an extra gene called the "enzyme functionality bit". This gene is a single letter. If its integer value is odd, the enzyme performs resource conversion in only one direction, whereas an even value indicates that the enzyme can convert resources in both directions. The conversion proceeds as follows. Given an enzyme, which converts between resources $a$ and $b$, let $r_a$ and $r_b$ be the amount of resources $a$ and $b$ in the agent’s reservoir. Let $x$ be a random number between 1 and $(r_a + r_b)$. If $x <= r_a$ then the enzyme will convert a unit of resource $a$ into a unit of $b$. If $x > r_a$ then the enzyme will convert a unit of resource $b$ into a unit of $a$, but only if the enzyme functionality bit indicates two-way conversion. Using the approach above, a two-way enzyme will tend to create equilibrium between the two resources, whereas a one-way enzyme will convert the source resource into the destination resource at a rate, which decreases with the amount of source resource available, making it less likely to rob the agent of all units of that resource.
3-9 Boundary configurations and issues

As mentioned previously, this model performs agent adhesion interactions, both when an agent reproduces and when neighboring agents are paired for normal interaction. These adhesion interactions proceed in a manner very similar to that described in [Holland] and in section 2-6. The main differences are as follows:

- Only single agents may be "engulfed". Multiple-agent aggregates and multiagents may not. The reason for this is simply that it seems subjectively unnatural to allow a single agent to engulf an aggregate containing potentially hundreds of other agents.

- Multiagents may not engage in adhesion interactions with agents outside their outer-most boundary. The reason for this is to ensure that the only agents within a multiagent are the agent compartments defined by their genomes and that those compartments are not removed from the multiagent by adhesion to external agents.

When it comes to adhesion interactions between neighboring agents, if the tag matching results in no adhesion, neither of the two agents are ejected from their aggregate. This ejecting only occurs when the adhesion interaction is between a mother agent and a newly created daughter agent.

When neighboring agents $x$ and $y$ engage in an adhesion interaction and $x$ is engulfed by $y$, $x$ is simply removed from whichever boundary it currently belongs to and is placed
inside a new boundary internal to the boundary of \( y \). The alternative, which is to place the entire aggregate, of which \( x \) is a part, inside the boundary of \( y \), seems too drastic. Similar rules apply when the adhesion interaction results in boundary sharing.

It is possible, either via adhesion interactions, taxation or random death, for all agents immediately located within a boundary, to be removed from that boundary. If this boundary still contains other internal boundaries, this results in what I shall refer to as boundary "bursting". When a boundary bursts, all of its internal boundaries are transferred into its parent boundary, as depicted in the figure below.

![Diagram of boundary bursting]

**Figure 27: Boundary "bursting"**

If the burst boundary has no parent boundary; that is, it is the outer-most boundary of an aggregate, then all of its internal boundaries are transferred to the site. It is possible for the site to be unable to accommodate the addition of these new aggregates. Those aggregates, which do not "fit" are considered to have been crowded to death. Their resources, both reservoir and genetic, are transferred to the site.
If the burst boundary is the outer-most boundary of a multiagent, the multiagent is killed. Its internal boundaries are transferred to its parent boundary, or to the site, if there is no parent boundary. All agent compartments within these internal boundaries are converted into regular agents and are tracked by the genome tracker. The surplus reservoir resources of the multiagent are distributed evenly among these new agents. This conversion and tracking is only performed on those compartments that will not be crowded to death. It seems pointless to track the existence of agents, which will neither survive until the next time cycle, nor engage in any interactions.

If the burst boundary contains no internal boundaries, it is simply removed from its parent boundary or its site. I refer to this as boundary "emptying". If the emptied boundary is a multiagent, it is killed and its reservoir and genome resources are transferred to the site. Note that, with the exception of combat deaths, whenever an agent is killed, its resources are transferred to the site. In the case of agent compartments within a multiagent, the resources are transferred to the multiagent's common reservoir and are considered part of the surplus (see section 3-3).

Figure 28: Boundary "emptying"
Most times, when the simulation is run, aggregation and formation of multiagents is negligible in comparison to the success of populations of lone agents. Multiagent populations rarely appear in the list of top ten genomes during a run. I believe part of the problem is a kind of bottleneck produced by the way aggregates and multiagents receive resources. A given site has a certain number of spaces allotted for agents, aggregates and multiagents to come and “feast” on the resources coming from the resource fountain. Agents, aggregates and multiagents each take up only a single one of these allotted spaces, regardless of their size in terms of internal agents, boundaries and so forth. Thus, a site full of lone agents can provide enough resources to sustain them all, whereas a site full of large aggregates or multiagents may only be producing enough resources to sustain one such aggregate, and those resources are being distributed over many. It would be useful to allow large aggregates and multiagents to span across several sites in proportion to the number of agents in their outer-most boundary, for example.

This bottleneck may be a factor contributing to the inability of multiagent populations to become more numerous than populations of lone agents. Another issue may be that ranking the top ten genomes by population alone is not an entirely appropriate measure of success, and that some form “biomass” calculation should factor into the ranking in some way.

The reason that aggregates have been permitted to form via adhesion interactions between neighboring agents is to provide a richer pool of aggregates from which to evolve multiagents. Without neighbor adhesions, agents will only aggregate in one of the
following three ways, with the exception of other configurations, which may arise via mutation.

1) **No aggregation**: Mother and daughter do not adhere to one another. Such a population has the potential to spread and vastly increase in number.

![Figure 29: Agents that eject their daughters](image)

2) **Layering**: Daughter or mother is engulfed. Such a population will not spread, except through boundary bursting, caused possibly by predation, in which case the population will most likely be devoured. There is also a severe bottleneck in such an aggregate. One agent must absorb enough resources for all the others.

![Figure 30: Agents that engulf their daughters](image)
3) **Swelling**: Daughter shares boundary of mother. Such a population will not spread, not even via boundary bursting. The population is doomed to remain within a single boundary and will be unable to grow much larger than the number of agents a single site can sustain. Short of a mutation, the only hope is to become a multiagent before the population is too large.

![Figure 31: Agents whose daughters share their boundaries](image)

**3-10 Mating and crossover**

For this model I have selected two-point crossover during mating. With one-point crossover, an agent with “good” genetic material at either end of its chromosome will be unable to pass on both of these traits to one of the hybrids that are produced. Two-point crossover does not suffer from this particular drawback.

As with other interactions in this model, the mating interaction is mediated by a tag matching calculation. The hamming distance is then used to calculate a mating probability. A perfect match results in probability 1 and no matching letter pairs will result in probability 0. If mating is left to proceed in this manner, crossover becomes rampant in the simulation, with new genomes being produced at such an astounding rate as to prevent any single genome from gaining a foothold. Another second effect of this
furious pace of genetic swapping is that the genome tracking data structure quickly swells to an incredible size. The resulting population log file rapidly attains sizes in the dozens of megabytes as well. Part of the problem is that, at the time of this writing, the system is not equipped to handle agent groupings at a higher level than that of specific genomes. There is no clustering into groups that might be likened to species, in the biological sense of the word. Because of this genome explosion, which cripples the simulation, the mating probability has been intentionally reduced by a significant factor. The probability described above is first squared and then divided by 100. Thus, even for agents with perfect mating tag and condition matches, the chance of performing crossover is a mere 1%.

To illustrate the problem, here is a stacked population graph from the log file of a typical 500-cycle run using the reduced mating probability described above.

![Stacked population graph of a typical 500-cycle run](image-url)

Figure 32: Stacked population graph of a typical 500-cycle run
Here is a second stacked population graph from an otherwise identical run, but without lowering the chance of mating.

![Graph showing population over time]

**Figure 33: A 500-cycle run without reducing the chance of mating**

The first run gave the following results after the 500th cycle:

- 117KB population log file
- 282KB of memory used by the genome tracking tree
- 1295 genomes produced in total
- 124 successful mating interactions
- 25 genomes present in cycle 500
- 1,064 individuals in the maximum population at cycle 500

The second run, however, had the following results:

- 5.3MB population log file
- 42MB of memory used by the genome tracking tree
• 161,405 genomes produced in total
• 105,335 successful mating interactions
• 2,952 genomes present in cycle 500
• 13 individuals in the maximum population at cycle 500
4. Testing

To determine whether or not the implementation constitutes a reasonable model for CAS, it must be demonstrated that it exhibits some common behaviors and phenomena that one would expect to witness in such systems. Presented in this section are six different CAS phenomena exhibited by the implementation. These are trade, cooperation, dominance through differential reproduction, extinction and two types of predation. At the time of this writing, the implementation is not equipped to detect other types of phenomena such as arms races or trophic networks (see [Holland] and [Hölldobler]).

4-1 Trade

The following three figures show agent populations that are trading with each other. In this population graph, and those that follow, various other pieces of information are graphed together with individual genome populations. These include genomic richness, which is the total number of genomes present, combat, which is the number of combat interactions per cycle, intergenomic trade, which is the number of trade interactions involving members of different species (provisionally defined to mean a group of agents with identical chromosomes) and intragenomic trade, which is the number of trade interactions between members of the same species. The combat and trade interactions counted are only those occurring between members of the top ten species during a given cycle. This eliminates possible noise contributed by potentially numbers species with extremely few members, making the graphs easier to interpret.
Figure 34: Trade example 1

In figure 34, the population, which takes 1st place some time near cycle 1190, appears to be trading with another of the top ten populations. The slight coincident bumps in trade and the 2nd largest population, between cycles 1500 and 1600, seems to indicate that this is at least one of the other partners in the trade.

Figure 35: Trade example 2
Figure 35 shows a second example of trade. The two populations, which are holding 1st and 2nd place between cycles 600 and 960, appear to be trading between one another. As the smaller population begins to decline beyond cycle 700, being overtaken around cycle 960, there is a similar decline in the amount of trade; evidence that it is this pair of species that are trading.

![Graph showing intergenomic trade and genomic richness over cycles 600 to 1200.]

**Figure 36: Trade example 3**

Figure 36 shows another example of trade. Beyond cycle 800, the largest population appears to be trading with the 2nd or 3rd largest, or both. Note that as the 2nd and 3rd largest species come onto the scene, between cycles 790 and 830, there is a corresponding jump in the rate of increase of the largest population. That, together with a coincident drop in the amount of combat, indicates that these species mutually benefit one another.
4-2 Cooperation

Cooperation is trade occurring between members of the same species. This can be seen in the following figures by observing the correlation between species populations and intragenomic trading.

![Graph with labels: intragenomic trade, genomic richness, intergenomic trade, combat.]

Figure 37: Cooperation example 1
4-3 Dominance through Differential Reproduction

In this example, the dominant species (genome 233) is overtaken by another (genome 349). The combat and trade graphs (barely visible) indicate that this is a case of differential reproduction only. Genome 349 is simply slightly more fit than 233 and gradually outnumbers it. A closer look at the data revealed that genome 349 arose from a single point-mutation in genome 233 altering one of the resources acted upon by the enzyme gene. As it turns out, this mutation made 349 less efficient than 233 at absorbing resources in proportion to the ratios in its chromosome. This seems very counter-intuitive until one considers that the extra resource surplus built up prior to reproduction, in 349, may act as a shield against taxation. Since 233 absorbs resources in a proportion that more closely matches the ratio of resources in its chromosome, it will build up less
surplus between reproductions and is thus more susceptible to being killed through taxation.

Figure 39: Differential Reproduction example

4-4 Extinction

Though rare, complete extinction has been observed in this implementation. The population decline shown in figure 40, below, eventually lead to complete extinction around cycle 18,000. The entire graph would span across six pages and looks little different than the right side of the graph below. Extinctions take a particularly long time due to freeloaders, which cannot be killed by taxation and must wait to be randomly killed.

Figure 40: Extinction example
4-5 Predation by combat

This is by far the most common CAS phenomenon observed in this implementation. It is marked by a tell-tale steep spike in the combat graph when one population overtakes another. This spike indicates that the second population overtakes the first by killing and devouring its members. Almost always, this attack is complete, wiping out the prey population entirely. This may be due to the inherent instability that comes with such small populations as discussed in [Levy]. It may also be due, in part, to predators that do not entirely rely on the prey for survival.

![Graph showing genomic richness and intergenomic trade over time.]

**Figure 41: Predation by Combat example 1**

Figure 41 shows three examples in rapid succession around cycles 6230, 6410 and 6520. The middle two populations barely have time to finish wiping out their predecessors before they too are being attacked.
Figure 42: Predation by Combat example 2

Figure 43: Predation by Combat example 3
4-6 Predation by Trade

The following example shows another type of predation in which the predator population wipes out the prey, not primarily through combat, but by robbing it of resources needed to survive.

Figure 44: Predation by Trade example
5. Conclusion

Though very abstract and minimalist in its framework, Holland’s Echo model leaves many design decisions and issues in the hands of the model implementer. Some of these issues are visible on the surface, while others do not become clear until design decisions force one to consider things more closely, such as the issue of how to handle boundary bursting in multiagents, or the balance between agent compartment reproduction and multiagent reproduction.

With Holland’s basic Echo model, including all five of its extensions implemented, it may now be possible to conduct potentially interesting experiments involving such things as the evolution of embroygenesis and other higher-level CAS phenomena.

5-1 What could have been done better?

In retrospect, there are a number of aspects and details of this model that could have been improved. It would be interesting to see whether or not the inclusion of a “wild card” resource in tag matching would greatly affect the kinds of evolutionary dynamics produced by the simulation. Certain aspects of the simulation that are fixed, such as the division of resources between a mother agent and its daughter, the replication factor of a multiagent or the maximum replication factor of an individual agent, could have been determined by genetics and put into the hands of evolution, instead of the programmer.
The resource bottleneck that often seems to prevent multiagents and aggregates from becoming more numerous could perhaps be overcome by giving aggregates and multiagents a "physical" size, preventing them from being outstripped by lone agents when it comes to resource absorption on sites. For that matter, including some measure of "mass" into the population ranking system might also give a more even-handed report of genomic success. Other measures of evolutionary success and dynamics might be better suited to a system like this, such as the classification of evolutionary dynamics used in [Bedau 98].

The genome tracking data structure might become more useful if it were to include such information as trading and combat interactions, to help determine the "why?" of a particular extinction, for example, instead of merely the "when?". It might also make more sense to cluster genomes into higher conceptual levels, such as species, thereby overcoming the crippling effects of the genome explosion when a "normal" mating probability is used.

5-2 Future work

My intention, with regard to future development, is to try to put Echo to work on some real world problem. The benefit of this may in fact be two-fold. By encoding the solution to some exogenous problem, such as sorting networks, for example, into the genetic code of agents and multiagents in Echo, and providing rewards in the form of resources, it may be possible to use Echo in the way standard genetic algorithms are used.
It should be noted that Echo would more than likely be far less efficient than a standard genetic algorithm, due to the computational overhead. However, this combination of implicit and explicit fitness may provide the kind of selection pressure needed to artificially accelerate the evolution of complex multiagent structures by encoding the problem in such a manner that lone agents, with only a single chromosome, are unable to seriously compete.

Another interesting direction for future work with the model may be to determine how, given the current model, the equivalent of classifiers may evolve. At the moment it isn’t clear how such phenomena can arise in Echo. Providing agents with some counterpart to a classifier “brain” may allow them to tackle harder problems, or exhibit more complex and life-like behavior. Such an augmentation of Echo should, ideally, have the property that aggregates and multiagents can exhibit behavior that is more than the sum of the parts, when it comes to their use of the “brains” of their agent compartments.
References


