An Introduction to SFI Echo.

Terry Jones [*] Stephanie Forrest [†]

November 16, 1993

[*]Santa Fe Institute, 1660 Old Pecos Trail, Suite A. Santa Fe NM 87501. email: terry@santafe.edu

[†]Dept. of Computer Science, University of New Mexico. Albuquerque NM 87131. email: forrest@cs.unm.edu

Contents

1 Introduction **1**
- 1.1 Echo . . . 1

2 Getting Started **2**
- 2.1 A Note On Fonts . . . 2
- 2.2 System Requirements . . . 3
- 2.3 Porting Echo . . . 3
- 2.4 Getting Echo Through Anonymous FTP . . . 4
- 2.5 Compiling Echo . . . 4
- 2.6 Preparing to Run Echo . . . 4
- 2.7 Running Echo . . . 5
- 2.8 A Sample Echo Run . . . 6

3 The Echo Interface **8**
- 3.1 The Overall Look And Feel . . . 8
- 3.2 The Control Window . . . 10
- 3.3 The World Editor . . . 13
- 3.4 The Site Editor . . . 15
- 3.5 The Agent Editor . . . 16
- 3.6 The Graphs . . . 17
- 3.7 Textual Output . . . 17

4 Creating A World **17**

5 Running A World **18**
- 5.1 Choosing A World To Run . . . 18
- 5.2 Editing a Running World . . . 19
- 5.3 Verbose Output . . . 19
- 5.4 Cluster Analysis . . . 19
- 5.5 Schema Tracking . . . 20

A Using FTP **21**

B Environment Variables and X Resources **23**

C Emacs Keybindings **26**

D Echo Widgets **27**

1 Introduction

This report is concerned with an implementation of a family of models of complex adaptive systems called Echo models. In what follows, you will find:

- An introduction to Echo.
- Information on how to obtain, install and run the Echo system.
- A description of Echo's graphical interface.
- Information on running Echo.

1.1 Echo

Echo is a model of complex adaptive systems formulated by John Holland [1, 2, 3]. It abstracts away virtually all of the physical details of real systems and concentrates on a small set of primitive agent–agent and agent–environment interactions. The extent to which Echo captures the essence of real systems is still largely undetermined. The goal of Echo is to study how simple interactions among simple agents lead to emergent high–level phenomena such as the flow of resources in a system or cooperation and competition in networks of agents (e.g., communities, trading networks, or arms races).

An Echo world consists of a lattice of sites. Each is populated by some number of agents, and there is a measure of locality within each site. Sites produce different types of renewable resources; each type of resource is encoded by a letter (e.g., "a," "b," "c," "d"). Different types of agents use different types of resources and can store these resources internally. Sites charge agents a maintenance fee or tax. This tax can also be thought of as metabolic cost.

Agents fight, trade and reproduce. Fighting and trading result in the exchange of resources between agents. There is sexual and non–sexual reproduction, sexual reproduction results in offspring whose genomes are a combination of those of the parents. Each agent's genome encodes various genes which determine how it will interact with other agents (e.g., which resource it is willing to trade, what sort of other agents it will fight or trade with, etc.). Some of these genes determine phenotypic traits, or "tags" that

are visible to other agents. This allows the possibility of the evolution of social rules and potentially of mimicry, a phenomenon frequently observed in natural ecosystems. The interaction rules rely only on string matching.

Echo has no explicit fitness function guiding selection and reproduction. An agent self–reproduces when it accumulates a sufficient quantity of each resource to make an exact copy of its genome. This cloning is subject to a low rate of mutation.

In preliminary simulations, the Echo system has demonstrated surprisingly complex behavior (including something resembling a biological "arms race" in which two competing agent types develop progressively more complex offensive and defensive combat strategies), ecological dependencies among different species, and sensitivity (in terms of the number of different phenotypes) to differing levels of renewable resources.

Ideally, Echo will allow the modeling of a diverse range of complex adaptive systems without the need for a specialized model for each to be developed. Typically, the people who know the most about any particular real–world complex adaptive system are not the people who can also develop sophisticated models that can be used as tools to increase understanding. Echo aims to provide a useful modeling tool or a starting point for the development of a model.

As a cautionary note, one must be a little careful when using the term "Echo." Properly, Echo refers to a large family of models. As described here, Echo will refer to the implementation developed at the Santa Fe Institute.

Several versions of the system have been developed by Holland, and there are significant differences between these. Echo has been described in [1, 2, 3]. These descriptions represent snapshots of ongoing thought about Echo models. The version implemented here is closest to that described in [1]. For further details, refer to the above sources.

2 Getting Started

2.1 A Note On Fonts

The following conventions are used consistently throughout this report:

- File names and UNIX commands appear in **bold face**. These will normally be references to files and commands that you might need or

use, not things you will be expected to enter immediately.

- Environment variables are shown in SMALL CAPS and are always completely UPPERCASE.
- Commands you are expected to type or put in a file are shown in sans serif. This is true even if what you are asked to type is a file name or an environment variable.
- Anything printed by the system appears in a *slanted* font.
- Things you will see in the Echo interface are typeset in a *slanted* font, as for other system output, but, additionally, are enclosed in "*double quotes*". The text so enclosed is exactly what you can expect to see in Echo's interface. For example, this font will be used when discussing the various options you'll see in Echo's popup menus.
- The UNIX shell prompt is a bold percent sign (%) at the start of a line.

2.2 System Requirements

Echo was developed on a SUN SPARC architecture running SUNOS 4.1.3. In theory it will run on other BSD–based versions of UNIX, but may require slight modification in some cases. The X Windows system from MIT is required. Development was under X11R5, but Echo should run without modification under X11R4. If you are using a SPARC machine, no special widget sets are required, the distribution file contains these. If not, you will need to **ftp** widgets from several locations – this is discussed in appendix D.

2.3 Porting Echo

Unless a joint research agreement with the Santa Fe Institute exists, there is currently no guaranteed way to get help porting Echo to a new architecture. A possible source of help is through electronic mail to **echo@santafe.edu**. Assistance will probably be contingent on the project being supported at SFI.

2.4 Getting Echo Through Anonymous FTP

Echo is available via anonymous ftp from **ftp.santafe.edu** in the file

/pub/Users/terry/echo/Echo-0.1.tar.Z

The entire distribution occupies just over two megabytes of disk space when uncompressed. You should transfer Echo, uncompress it and extract the files using tar. If you are not familiar with the workings of ftp, uncompress or tar, consult appendix A.

2.5 Compiling Echo

Once you have received and unpacked the distribution, you will need to compile or "make" it. The **make** command will take care of the compilation and linking, but first you will need to make some changes to the file **Makefile**. These should be very minor, the file contains instructions to help you. You should only have to change two lines, to indicate where the X libraries and include files are on your system. If you don't know, you can try to find them or ask your system manager.

Once you have done this, you can simply type

```
% make all
```

and make will do the rest. You should see about twenty files get compiled. When the make is completed, you should have a file called **Echo** which is the executable program. This process also creates you a shell script, **run–echo** to make it simple to run the executable in the recommended way.

2.6 Preparing to Run Echo

Two additional things need to be done before running Echo.

- X resources for Echo need to be installed.
- Shell environment variables need to be set.

The distribution contains programs that will do this for you if do not want to do it yourself, or do not know how. If you'd like it taken care of for you, use

```
% make simple-setup
```

This will add X resource defaults to your X startup file and set environment variables in your shell's startup file. If you have used this option, you should now log out and then log back in, as these changes will not affect your current login session[1].

If you'd prefer to do this kind of installation yourself, you should see the details about what needs to be done in appendix B.

2.7 Running Echo

Once you have completed the above, you should be able to run Echo. Echo writes its textual output to the **xterm** from which it was invoked. The X resources you just installed specify the colors, sizes and locations of all the pieces of the Echo interface. These pieces are designed to fit around a central **xterm** window.

There are two ways you can arrange things so that an **xterm** is created in the right place (and the right color):

- The simplest way is to use the shell script **run-echo** that was made for you when you did the original **make**. This will do all the work of setting up an **xterm**, invoking Echo, and making sure the interface is configured correctly.

- Slightly more work, but simpler in the long run is to put Echo into a popup menu. This is achieved by adding a line to your window manager's initialization file. This file's name depends on your window manager. A likely location is **~/.twmrc**, or some other file in your home directory whose name starts with a period (.) and ends in the string "**wmrc**". If you don't know where your window manager initialization file is or can't find it, ask your system manager. The file **.window_manager** in the Echo distribution contains a line that can probably be usefully inserted in this file[2].

[1] They will be automatically set for you in all future login sessions.

[2] **Warning**: it is not as simple as just adding the line to the end of this file. If you are unsure about what you're doing, get help.

2.8 A Sample Echo Run

This section walks you through a quick Echo demo, without too much explanation.

Once you have installed the X resources and the shell environment variables, you are in a position to run Echo. Try it now – either from your window manager or with

```
% ./run-echo
```

You should see a number of windows appear on your screen. These are explained in the next section. For now notice the main menu bar at the top of the screen and the **xterm** in the middle.

- The first step in running a world is to choose the world you intend to run. To do this, click on the red menu button at the top of the screen that says "*Running*". Hold the mouse button down and drag the pointer down to "*Choose World*" and let go.

- A file selector will pop up. This allows you to walk through the directory structure to find a file containing the specification of a world you want to run. You should see "insects" in the scrollable region on the right. Click on this word and the line it is on should change to white on black. Now click on "*OK*" to select the world.

- Look at the "*Species*" graph. It now has a legend, showing "*ant*", "*fly*" and "*cat*" ("cat" is short for caterpillar). The legend has appeared as the world you selected contains a site that contains all of these agent types.

- Now select "*Run 10 Generations*" from the "*Running*" menu. You should see the two graphs update and the **xterm** will show the text: "*Run through generation 10.*"

- Let's edit the world and remove a few agents. Click on the "*Edit*" menu button and drag the pointer down to "*Stack*". A window will appear showing you the population of agents present at the site. This is called the "stack" at that site. It's a one–dimensional array of agents. Use the left mouse button to move the small arrowed cursor to the start of

some line. Now hit Control–k twice on your keyboard. You should see the line in the stack disappear – and with it some unfortunate agent. Control–k is an Emacs keybinding that kills the text until the end of the line. See appendix C for more information on the keybindings if these are not familiar to you.

Now that you've removed the agent (or agents if you were feeling enthusiastic), hit the "*Amen*" button at the top of the stack editor to make it so.

- Before restarting the world, choose "*Set Verbose Level...*" from the "*Control*" menu at the top of the screen. Here you can type letters to see certain types of output in the **xterm** window. In the small gray window at the bottom of the panel that popped up, type the following incantation: "gsukbx". You can see what output each of these letters will produce by reading the text above the small text window. To type in the text window you'll need to move the mouse into it. Now click on the "*OK*" button to dismiss the window.

- Choose "*Run 1 Generation*" from the "*Running*" menu. You should see various output appear in the **xterm**. This will likely include information on the agents that were taxed during the generation and those that went bankrupt. There is a small chance you'll see a mutation or an agent killed in combat. The end of the output will show a population summary.

- Now we'll run to generation 100. Choose "*Run Until...*" from the "*Running*" menu. In the small window that pops up, enter 100 and then click on "*OK*". You should see the graphs rapidly update to generation 100. You'll also see a lot of text flashing by in the **xterm** window. Often it is good to turn off all such output when you are about to run for a large number of generations.

- Select "*Variant Levels*" from the "*Graphs*" menu. The graph will appear where the "*Worldwide Population Level*" graph was (actually it's just on top of it). The top line (in red) shows the number of different genomes that have ever existed in the world. The bottom (blue) line shows the number that are currently alive. You can compare this number with the worldwide population level, and it will probably be

quite a lot smaller. Obviously, some genomes have multiple copies – presumably because they are doing something right.

- Finally, choose "*Cluster Living Population*" from the "*Examine*" menu. You will see a tree displayed in the **xterm** window that indicates how closely related the agents in the world are. You will probably need to resize (or scroll) your **xterm** window to see the whole tree. This tree will be explained in more detail later.

- To exit Echo, choose "*Exit*" from the "*Control*" menu. Of course you can continue to play if you like.

The next section gives more details about the kinds of things you just saw and did.

3 The Echo Interface

After starting Echo, you should see seven windows on your screen. Starting in the center at the top and moving counter–clockwise, these are the Control window, the World Editor, the Site Editor, the Agent Editor, a graph showing species levels, a graph showing the worldwide population level, and an **xterm** window.

3.1 The Overall Look And Feel

There are some high level features of the interface that have been designed to make the look and feel consistent. These are:

- In general, the interface makes no distinction between the buttons on your mouse (assuming you have more than a single button). You can use any button you like to push command buttons, or use popup menus. The only exception to this rule is described in the next point.

- Some of the areas of the screen are a steely grey color. These are text windows. This color is used consistently to indicate an area of the screen where you may type characters. Text in these windows will always be black. All you need do is move the cursor into the grey

area and start to type. Each of these is in fact a small editor with Emacs–like keybindings[3].

In a text window, the mouse may be used to copy and paste regions in the same fashion as in normal **xterm** windows. The first button, when held down, can be used to drag out, and thus copy, a region (which will be highlighted). The second button pastes the copied text into the buffer at the cursor's location. The third button can be used to extend the copied region. This is something you should experiment with if you are not already familiar with this kind of mouse behavior.

In addition, rapid double and triple clicks with the left mouse button can be used to select the word the cursor is on or the line the cursor is on. A single click with the first button just moves the text cursor to the location of the mouse.

A search window can be popped up by typing Control–S, and the contents of a file can be read into the buffer using Meta–I, which pops up a window to read the file name. See the Emacs keybindings in appendix C for more information on operations in text windows.

- Blue areas where white text appears, such as the main output window, are always read–only.

- Command and menu buttons always have a red background and white text. These always indicate things you can click the mouse on to have some action performed. Command buttons carry out an action immediately, while menu buttons pop up a menu of options. As mentioned above, no distinction is made here between the buttons on your mouse – use whichever you like.

- There are a number of occasions when Echo will need to get a file name from the user. To do this, it will display a file selector window. This window has various components. You may enter the file's name in the text window at the top of the file selector, and then click on "*OK*". Often simpler, is to use the mouse to select the file you want.

[3]If you are not familiar with Emacs, don't worry, you can move around with mouse button 1 and use the delete key to get rid of things you don't want. Appendix C gives a brief introduction to editing with Emacs.

The scrollable window labeled "*Directory Contents*" contains the list of files in the currently scanned directory. Echo tries to make sure that the default directory is a useful one (using the ECHO_LOCATION environment variable).

You can select a file in the scrollable list by simply clicking on it with the mouse. The second mouse button can be used to scroll the list. If you click on the entry labeled "../" the file selector will read and display the contents of the parent directory. The "*Up*" button can also be used to move up a level. If you click on an entry that is a directory, the file selector will read and display that directory's contents. Directories can be identified by the trailing "/" after their name.

The "*OK*" button selects the file and causes the file selector to disappear. The "*Cancel*" button dismisses the file selector and cancels whatever function was called that made it appear originally.

3.2 The Control Window

The control window consists of five pull down menus. These are labeled "*Control*," "*Edit*," "*Running*," "*Graphs*," and "*Examine*."

- The "*Control*" menu contains three options. The first, "*Set Verbose Level...*" pops up a window that contains lines that briefly describe types of output that can be displayed in the main **xterm** window. At the start of each of these lines is a key letter. In the small grey window at the bottom you may enter the letters corresponding to the types of output you want to see in the **xterm**. This output will appear following each generation, so you must run at least one generation to see anything.

  The "*Show Seed*" option of the "*Control*" menu prints the current random seed value into the **xterm**. This allows you access at any time to the seed that was used to set up the current run.

  The "*Exit*" option does exactly that, it exits Echo completely. It does not currently ask for confirmation, so be careful!

- The "*Edit*" menu has seven options. Most of these are very similar. "*Worlds*" will pop up a file selector to let you choose a world to edit.

The file selector can be used to wander through a directory tree and select a file containing the specifications of a world. Use the mouse to click on the name of the file you want (or enter it in the top grey window) and then click the "*OK*" button to select it. The world you choose will appear in the World Editor window in the top left corner of the screen (more on that soon).

Choosing "*New World*" will clear the World Editor window and let you enter the details of a fresh world.

The options "*Sites*," "*New Site*," "*Agents*," and "*New Agent*" all behave in a similar fashion. They can be used to prepare for editing the characteristics of sites and agents in the Site Editor (middle left) and Agent Editor (bottom left) windows.

The final option in the Editing menu is "*Stack.*" This allows you to edit a site in a running world. Since we have not yet begun to see what happens when a world is actually running, this operation will be described later.

- The "*Running*" menu has nine options. The first, "*Choose World*" is used to tell Echo which world you intend to run. This option must be used before an Echo run can begin. It also pops up a file selector so that you can choose a world.

  The next four options, "*Run Indefinitely,*" "*Run 1 Generation,*" "*Run 10 Generations,*" and "*Run Until...*" are all to do with running a world for a certain time. They should all be self explanatory. The last will pop up a window so you can enter a stopping generation number.

  The next two options, "*Pause*" and "*Continue*" can be used to temporarily halt and restart a run. If you have used one of the above running options to run until generation 500 and suddenly decide you need to turn off the output in the main text window, you can use "*Pause*", turn off the output and then "*Continue*" to allow the run to proceed to generation 500. These two options can be used in this manner any time Echo is running an experiment.

  The "*Replay*" option resets the world and re–initializes the random number generator so that the run can be re–done exactly. This is very

useful when you would like to try some experiment and need to stop the world at an earlier point.

The "*Seed*" option can be used to enter the random seed for a run. The seed should be set *before* you choose the world to run. It is important that you perform these two in this order. If you do not specify a random seed for the run, one will be chosen for you. The seed chosen for you is guaranteed to be unique, and the random number generator has been highly scrutinized for randomness. The file **random.c** contains a blow–by–blow description of the search for an acceptable generator.

- The "*Graphs*" menu can be used to pop up any of five graphs. Initially there are two places where graphs appear, and two of the five are shown by default. The species level graph always appears on the left while the other four are on its right. Of course, since these are normal X windows, you can move them around and resize them as you wish. The five graphs are:

  - The "*Species Levels*" graph shows a legend (after you choose a world) and plots the population level of all the descendants of the original members of the "species" . It is not really correct to call these groups species (lineage is more accurate), but I will not go into that here.
  - The "*Worldwide Population Level*" graph shows the total number of agents alive in the world.
  - The "*World Resource Levels*" graph shows how many of each resource exist (this is often very dull viewing).
  - The "*Schema Level*" graph allows you to track the level of a schema in the population.
  - The "*Variant Levels*" graph shows the number of genomes that have ever existed and the number of genomes that currently exist.

- The "*Examine*" menu allows you to look at some properties of the populations. The first option "*Choose Schema to Graph...*" allows you to enter a regular expression corresponding to a schema you are interested in. The regular expression is in UNIX–style and is a pattern of resources that might be found on an agent's genome. The simple details

of these expressions are explained in section 5.5. More information on regular expressions can be found in the UNIX manual page for **egrep**.

The next two options, "*Cluster living population*" and "*Cluster all individuals*" both perform cluster analysis. You may choose to have either the current population clustered or every genome that ever existed clustered. The output will appear in the **xterm** window, which should have a scrollbar in case the output is too long. Clustering is explained in section 5.4.

3.3 The World Editor

The "*World Editor*" is used to make changes to the properties of a world. These changes do not affect the currently running world. You can choose the world you wish to alter (or a new world) from the "*Edit*" menu in the "*Control*" window. Worlds have thirteen properties. Eleven of these are edited by entering their values into the grey horizontal text windows to the right of their brief descriptions:

- Each world has a "*Name*". Usually this is the same as the "*File Name*" in which the world is stored, but this need not be the case. The file name is the name of the file in the "*OBJECTS/worlds*" directory pointed to by the ECHO_LOCATION environment variable. There can only be one world per file, but many worlds (stored in files with different names) may have the same name.

- Each world has some "*Number Of Resources.*" The resources are named a, b, c... depending on the number that exist. Typically this is a small number, say three or four.

- The "*Rows*" and "*Columns*" give the size of the two dimensional array of sites.

- The "*Trading Fraction*" determines how much of the excess of a resource an agent gives away when it trades. Each agent trades a particular resource, and when it gets involved in a trading relationship with another agent, it uses this fraction to decide how much to give away. The excess is defined to be the amount of the resource in question over and above what the agent needs for self–replication purposes.

- The "*Interaction Fraction*" determines how many agent–agent interactions will take place each time step. This number is multiplied by the population size to arrive at a number of interactions. After this many interactions are performed, the sites produce resources again and the other aspects of an Echo cycle are executed.

- The "*Self Replication Fraction*" determines how much of its extra resources a parent will give to a child when self replicating. Self replication involves making a copy of one's genome when enough resources have accrued in the reservoirs. Once this happens, there may be extra resources, some of which might be given to the child to ensure that it is not too weak at birth.

- The "*Self Replication Threshold*" determines how many copies of its genome an agent must be capable of making before it actually makes a single one. Thus if this value is set at 2, the agent must accumulate twice as many of each resource in its reservoirs as it has in its genome. In this case, when it does make a copy of itself, it will have an excess of each resource equal to what it carries in its genome. This extra can be divided between the agent and the new child according to the Self Replication Fraction described above.

- The "*Maintenance Probability*" determines the frequency with which sites charge a maintenance fee. This probability is used per site per agent per Echo cycle.

- The "*Neighborhood*" can be set to any of "NONE", "EIGHT" or "NEWS" to indicate how agents can migrate in the world. The first should be clear, it disables migration. The second means an agent can move to any of the eight adjacent sites (assuming it is not in a corner or on an edge) and the third allows only north, south, east or west moves.

In addition to these eleven properties, worlds have an array of **Sites** and a "*Combat Matrix*." These can both be edited by clicking the button at the top of the World Editor, which will cause a window to pop up. The sites window should contain site file names (see below) in an array the size of the Rows and Columns as specified in the world's properties above. The combat matrix is a square array of side length the number of resources in the

world. Its actual use is somewhat complicated and will be described more fully below under Combat.

3.4 The Site Editor

Sites have ten properties, nine of which can be entered directly into the Site Editor window in the horizontal grey rectangles:

- As with worlds (and agents), sites have a "*Name*" and a "*File Name*" in which they are stored. The actual location of the site file is in the "*OBJECTS/sites*" directory.
- Each site has a "*Mutation Probability*." Mutation is performed in a genetic algorithm fashion. Each locus on each genome is mutated with this probability at the end of each cycle. The allele at a locus may "mutate" to the same allele value.
- The "*Crossover Probability*" determines the probability that crossing over, or recombination, takes place when two agents reproduce sexually. If recombination does not take place, the agents are left untouched.
- The "*Random Death Probability*" is the probability that an agent is killed without cause at the end of a cycle. This is typically set very low. After each cycle, every agent is killed for no reason with this probability.
- The "*Production Function*" determines how much of each resource the site produces at the end of each cycle. These values should be specified separated by white space. There should be as many of them as there are resources. The same is true for the next three properties.
- The "*Initial Resource Levels*" are the resource levels that the site is allocated when the world is initially created.
- The "*Maximums*" determine to what level each resource can grow if it remains "on the ground" (i.e. not picked up by an agent) at a site.
- The "*Maintenance*" is the tax charged by the site. Each agent is charged this tax after every cycle according to the maintenance probability set for the world. This probability is used to determine whether

each agent individually gets taxed, not whether the site will charge all agents if the probability condition is met.

The final property of a site is its initial agent list. This can be accessed by clicking the mouse on the "*Agents*" button at the top of the site editor. A window will pop up. Each line in this window is used to specify some number of agents. The agent names must be agent file names. Each name may be followed by white space and a decimal number indicating the number of agents of that type that should be created contiguously at that location. The agents in this list form the agent stack at that site.

3.5 The Agent Editor

Agents have eleven properties:

- As with worlds and sites, agents have both a "*Name*" and a "*File Name*." The file name simply references a file in the directory "*OBJECTS/agents*."
- Each agent has a "*Trading Resource*." This is the resource that the agent, initially, trades. This may be mutated in the course of a run.
- Each agent is provided with some "*Initial Resources*." This specifies the resource levels in the agent's reservoir when it is created. There should be as many numbers here as there are resources, each separated by white space.
- The "*Uptake Mask*" determines what resources the agent is able to pick up directly from the ground at the site. This should be a string of "1" or "0" characters, one for each of the resources. They should not be separated by white space. A "1" indicates that the agent may pick up this resource, and a "0" that it may not. This mask is subject to mutation.
- The next six properties all specify tags and conditions. These are used to determine with whom and how the agent interacts in the world. They are described in detail elsewhere [1, 2]. These "genes" can all grow and shrink (even to zero length) under mutation.

3.6 The Graphs

There is not much to say about the graphs. The individual graphs are briefly described in section 3.2. There is space allocated for two of them, side by side at the bottom of the screen. Only the species graph is ever displayed on the left. However, since they are fully functional X windows, you can use your window manager to resize and reposition them as you wish. The exit button on each graph window simply closes the window. Graphs can be redisplayed by selecting the graph in question from the Graph menu in the Control window. The graphs (currently) update every two hundred generations and there is no way to retrieve data once it moves off the left of the graph (other than by replaying the world). This will hopefully be changed sometime.

3.7 Textual Output

Textual output appears in the **xterm** window. This window should have a scroll bar so you can examine lengthy output. The **–sl** option to **xterm** can be used to set the number of lines that are saved off the top of the screen for scrollback purposes. If you invoke Echo with the supplied **run–echo** script (or by adding the suggested line to your window manager's startup file), the **xterm** created will have a scroll bar and will save two thousand lines of previous text.

4 Creating A World

The Echo distribution comes with several worlds, sites and agents in the **OBJECTS** directory. You should never need to directly edit the files under this directory, the world, site and agent editors are designed to read and write these for you.

These editors are described in sections 3.3, 3.4 and 3.5.

To create a world (including its sites and their agents) from scratch, choose "*New World*" from the "*Edit*" menu. This will display a blank world editor. Fill in the details of your new world, including a name and file name, and then "*Save*" it. Notice that you need to fill in the sites array. Click on the "*Sites*" button to display a text window in which you enter the site names (actually the site file names). This should be an array that has as many rows and columns as you specify in the world editor. For an example,

take a look at the world **4x4–insects** in the Echo distribution. This is a world with four rows and columns. Its site array specifies the same site 16 times (the site is also called **4x4–insects**.

Another way to create a new world is to copy an existing one. This is easily done. Suppose you wish to make a copy of the **insects** world. Read it into the world editor and change its name and file name. Then make the other changes you want and save the new world. The same principle applies to making copies of sites and agents.

Note that worlds refer to sites (in the "*Sites*" text window of the world editor) and that sites in turn refer to agents (in the "*Agents*" text window of the site editor). This is obvious, but the fact that it implies a connection between the three editors you'll be using may not be so clear. If you are unclear about how to create world, sites and agents, the best way to look is to examine the ones in the Echo distribution.

If you don't own the Echo distribution files, you will not be able to save your creations in the **OBJECTS** directory that the distribution came with. There is a simple solution to this: simply copy the distribution's **OBJECTS** directory elsewhere, and change your ECHO_LOCATION environment variable to indicate where your personal Echo objects are to be found. For example, if the Echo distribution is located in **/usr/local/Echo**, you can create your own objects directory in your home directory with

```
% cd
% cp -r /usr/local/Echo/OBJECTS Echo-objects
```

and then change ECHO_LOCATION to be the **Echo–objects** directory in your home directory.

Of course, you don't need to copy the distribution's entire **OBJECTS** directory, you can just create your own. Echo expects the directory specified in the ECHO_LOCATION variable to contain three sub–directories, named **worlds**, **sites**, and **agents**.

5 Running A World

5.1 Choosing A World To Run

The first step in running a world is choosing which world to run. To do this, select the "*Choose World*" option from the Running menu in the Control

window. This pops up a file selector showing the available worlds. Once you choose a world, you can use the various entries in the "*Running*" menu to actually run it.

5.2 Editing a Running World

Although you cannot yet change attributes of sites and worlds while a run is in progress, you can make changes to the agents at a site. If you choose "*Stack*" in the Edit menu, a window will pop up containing the agents present at the site (you will be asked to enter site coordinates if you have multiple sites). In this window you can directly edit the genome of any agent. You can search for a particular string (use Control–s), you can read in a file of agents from disk (use Meta–i) or you can use the editor to remove or replicate some number of agents. Once you are done, you can use the "*Amen*" command button to make your new site reality. The help button will pop up a box describing the genome representation.

5.3 Verbose Output

The "*Set Verbose Level...*" option in the "*Control*" menu can be used to make informative text appear in the X window. This has already been described above. Interesting output includes that for the letters `g` (generation number), `s` (species summary), `u` (details of mutations), `k` (who is killing whom), and `d` (to see extinct genomes).

5.4 Cluster Analysis

It is possible to perform a cluster analysis based on the genetic distance between the genomes. This can be done for the living genomes or all those that have ever lived. The output will appear in tree form in the **xterm** window.

The clustering is done by calculating the genetic distance between all pairs of agents. This distance is defined as the minimum number of mutations needed to transform the genome of one agent into the genome of the other. The two agents that are most closely related are grouped into a "cluster". By also defining the distance from an agent to a cluster and between separate clusters, it is simple to build a tree showing how closely related the individuals

in a population are. This is done by successively merging the closest clusters (or individuals) into a larger cluster until only one remains. This process naturally defines a hierarchy of cluster relatedness which can be displayed as a tree. The clustering algorithm runs in $O(n^3)$, where n is the number of individuals at the outset, so you may have to be patient if you have a large population.

5.5 Schema Tracking

While the world is not running, you can enter a schema to graph by selecting that option in the "*Examine*" menu. Genes are separated on the chromosome by an underscore (_). The meta–characters ^ and $ can be used to tie the regular expression to the beginning and end of the genome respectively. Square brackets ([]) can be used to denote a set of characters any one of which constitutes a match. A star (*) represents any number of the preceding expression and a plus (+) represents one or more of the preceeding expression. All this is very standard regular expression syntax, and this explanation is meant to be brief at best.

As an example, we could look for agents that had the string "aa" somewhere in their mating tag[4]. This is matched by the expression
^[abcd]*_[abcd]*_[abcd]*aa

which allows anything (including nothing) in the first two genes and then anything (including nothing) followed by two a's in the third gene. The range "[abcd]" could also have been represented with "[a–d]".

To see the level of this schema in the population, select the Schema Level graph in the Graphs menu.

[4]Assume a world with four resources.

Appendices

A Using FTP

The following illustrates how the file may be retrieved. Your UNIX prompt is a percent sign. What the system prints is shown in a *slanted* font, and what you type, as usual, is in sans serif.

% ftp santafe.edu
Connected to santafe.edu.
220- * * *

230- please send email to <ftp@santafe.edu>.
230-
230-Please read the file README.Z
230- it was last modified on Fri Mar 19 15:29:44 1993 - 179 days ago
230 Guest login ok, access restrictions apply.
ftp> cd pub/Users/terry/echo
250 CWD command successful.
ftp> binary
200 Type set to I.
ftp> get Echo-0.1.tar.Z
200 PORT command successful.
150 Opening BINARY mode data connection for Echo-0.1.tar.Z (925969 bytes).
226 Transfer complete.
local: Echo-0.1.tar.Z remote: Echo-0.1.tar.Z
925969 bytes received in 6.1e+02 seconds (5.2 Kbytes/s)
ftp> quit

Now you have retrieved the entire distribution. The distribution consists of a number of files and directories, which were archived into a single file that was then compressed. The next job is to reverse these steps to recover the original Echo files. To do this:

```
% uncompress Echo-0.1.tar.Z
% tar xf Echo-0.1.tar
```

This should result in the creation of a new directory, called **Echo-0.1**. Check that that directory has been created. If so, and you have received no error messages, it is safe to remove the bundled distribution file with

```
% rm Echo-0.1.tar
```

B Environment Variables and X Resources

This appendix deals with setting up environment variables and X resources for those who preferred not to have this done automatically in section 2.6.

You may wish to automatically install just the X resources, or just the shell's environment variables. Do this with either

```
% make x–setup
```

or

```
% make sh–setup
```

In both cases, the shell script that is invoked (either **echo–x–setup** or **echo–sh-setup**) tries to find an appropriate file to append some text to. It is fairly conservative and always makes a backup copy of any file it alters (in a file whose name ends with **.bak**).

There are several reasons why you may choose not to have either of these setups done automatically. If you are an experienced UNIX user, your shell startup and X resource files are probably not something you'll feel comfortable having someone else's shell script edit automatically. In this case, you should be able to decide how you wish to do what follows.

Here are more details about exactly what needs to be done:

- The file **Echo.ad** contains X window resource specifications. These set the various colors of the Echo interface, set the sizes and locations of the pieces of the interface and so on. If you know what all this means, you should put the contents of this file somewhere that the X toolkit will find them when Echo starts.

- Echo looks for certain environment variables that can be used to influence its behavior. None of these are required, but at least one is highly recommended.

  Echo comes with a collection of pre–defined worlds, sites and agents which are located in the **OBJECTS** directory. If you have installed Echo in the directory **/usr/local/echo** then you should set an environment variable called ECHO_LOCATION and give it the value

/usr/local/echo/OBJECTS

If you do not do this, you will always have to invoke Echo from the directory where you installed it for it to see the **OBJECTS** directory. Eventually you may have your own directory of Echo objects and you can change this variable to point to it.

The way to set the environment variable depends on the shell you are using. If you are in csh, the simplest thing to do is to place the line

```
setenv ECHO_LOCATION /usr/local/echo/OBJECTS
```

in your ~/**.login** file. Then, whenever you log in, the variable will be automatically set for you. To make it affect the current login session, you can also type the line at the shell's prompt.

If you are not using csh or a variant of it, you should place the following lines in your ~/**.profile** file (or equivalent),

```
ECHO_LOCATION=/usr/local/echo/OBJECTS
export ECHO_LOCATION
```

And you can either log out and in again or type those two lines to the current shell to have them affect this session.

If you aren't sure what shell you are running, type

```
% echo $SHELL
```

and if the output ends in the letters *csh*, then use the first method above. If not, use the second[5].

The other three environment variables are less important, but will make Echo start in a more attractive fashion. The variables and their recommended values are given below. These settings will make Echo display a world, a site and an agent when it starts up. Otherwise the three editing areas will be blank.

[5]If the shell name ends in *bash*, the lines should go into your ~/**.bash_profile** file if you have one and your ~/**.profile** if not.

These variables can be set in exactly the same way that you set the variable ECHO_LOCATION above. If you use C shell or a variant of it, the following goes into your ~/**.login**:

```
setenv ECHO_WORLD insects
setenv ECHO_SITE insects
setenv ECHO_AGENT fly
```

Otherwise, the following goes into your ~/**.profile**:

```
ECHO_WORLD=insects
ECHO_SITE=insects
ECHO_AGENT=fly
export ECHO_WORLD ECHO_SITE ECHO_AGENT
```

C–a	Beginning Of Line.	M–b	Backward Word.
C–b	Backward Character.	M–d	Delete Next Word.
C–d	Delete Next Character.	M–f	Forward Word.
C–e	End Of Line.	M–i	Insert File.
C–f	Forward Character.	M–v	Scroll Backwards.
C–h	Delete Previous Character.	M–<	Beginning Of File.
C–k	Kill To End Of Line.	M–>	End Of File.
C–l	Redraw.		
C–n	Next Line.		
C–p	Previous Line.		
C–r	Search Backwards.		
C–s	Search Forwards.		
C–t	Transpose Characters.		
C–v	Scroll Forward.		
C–w	Delete Selected Region.		
C–y	Paste Deleted Region.		

Table 1: Useful Emacs Keybindings For Text Windows

C Emacs Keybindings

In this section an uppercase C will be used to represent the use of the Control key and an uppercase M to represent the Meta key. Thus, C–x indicates that you should hold down the Control key and while doing so, type an x. The Meta key is used in an identical fashion, e.g. M–d tells you to first hold down Meta and while holding it, press the d key.

Table 1 shows the most useful key bindings present in all text windows.

D Echo Widgets

If you are not installing Echo on a SUN SPARC machine, you will need to obtain and install various widget sets on your machine before you can make Echo. These are all freely available via **ftp**. If you are not familiar with this sort of installation, it might be best to consult your system administrator.

The widget packages all create libraries that Echo must be linked with when it is compiled. The **Makefile** definition of **LIBS** in the distribution assumes that these libraries can be found in the **WIDGETS** directory. In fact, these libraries can be installed anywhere that is convenient, as long as the **Makefile** is altered to reflect their location.

The various widget libraries and their **ftp** locations are as follows:

- **Athena Plotter Widgets**
  Machine: **ftp.uni–paderborn.de**, Location: **/unix/tools**.
- **Free Widget Foundation Widgets**
  Machine: **a.cs.uiuc.edu**, Location: **/pub/FWF**.
- **3D Athena Widgets**
  Machine: **ftp.x.org**, Location: **/contrib/Xaw3d**.

Each of these widget sets comes with instructions on how to make and install the libraries that support the various widgets. The FWF and Xaw3d widgets are available for **ftp** from many locations. Use **xarchie** to find other sites.

Once these libraries have been created and Echo's **Makefile** knows where to find them, you should be able to proceed with the Echo installation itself.

I apologize if you have to go through this procedure. It is the result of using the X toolkit and the need to find useful (free) widgets. I hope this will be solved in later Echo versions by using **TCL** – though this will require that the machine receiving Echo has (or obtains) the **TCL** libraries...

References

[1] Holland, John H. (1992a). *The Echo Model*, in "Proposal for a Research Program in Adaptive Computation". Santa Fe Institute, July 1992.

[2] Holland, John H. (1992b). *Adaptation in Natural and Artificial Systems*, 2nd Ed., MIT Press, 1992.

[3] Holland, John H. (1993). *ECHOING EMERGENCE : Objectives, Rough Definitions, and Speculations for Echo-class Models*, To appear in "Integrative Themes", George Cowan, David Pines and David Melzner Eds. Santa Fe Institute Studies in the Sciences of Complexity, Proc. Vol XIX. Reading, MA: Addison–Wesley 1993.

Index

Agent Editor, 16
Agent File Name, 16
Agent Name, 16
Agent Reservoirs, 16
Agent Resource Levels Initially, 16
Agent Uptake Mask, 16
Agents List At A Site, 16

Choosing A Schema To Graph, 12, 20
Choosing A World, 6, 11, 19
Cluster Analysis, 8, 13, 19
Columns, 13
Combat Matrix, 14
Command Buttons, 9
Conditions, 16
Continuing A Paused Run, 11
Control Menu, 19
Control Window, 10
Copying Agents, 18
Copying Sites, 18
Copying Worlds, 18
Creating A World, 17
Creating New Agents, 11
Creating New Sites, 11
Creating New Worlds, 11
Crossover, 15

Dead Agents, 19
Demo Run, 6

Echo Interface, 8
Echo Look and Feel, 8
Echo Objects, 13, 15, 16, 23
Echo Resources, 13, 16
Echo Widgets, 27
Editing Agents, 11
Editing Menu, 10
Editing Running Worlds, 19
Editing Sites, 11
Editing The Stack, 6, 11
Editing Worlds, 10
Emacs Keybindings, 26
Environment Variables, 23
Examine Menu, 8, 12
Exiting Echo, 8, 10
Extinct Genomes, 19

File Selectors, 9
Fonts, 2

Generation Number, 19
Graphing A Schema, 12, 20
Graphs, 17
Graphs Menu, 12

Initial Site Resource Levels, 15
Interaction Fraction, 14

Maintenance, 14, 16
Making Echo, 4
Maximum Resource Levels, 15
Menu Bars, 9
Migration, 14
Mouse Buttons, 8, 9
Mutation, 15, 17, 19

Neighborhood, 14
Number Of Resources, 13

Output, 19

Pausing A Run, 11
Pop–up Menus, 3, 5, 6, 8
Population Level Graph, 12
Porting, 3

Random Death, 15
Random Seeds, 10, 12
Read Only Output, 9
Recombination, 15
Regular Expressions, 12, 20
Replaying A Run, 11
Resource Levels Graph, 12
Resource Levels Initially, 15
Resource Levels Maximally, 15
Resource Production, 15
Rows, 13
Running A World, 19
Running Echo, 5, 6, 11
Running Menu, 6, 7, 11

Sample Run, 6
Schema Levels Graph, 12
Schema Tracking, 20
Self Replication Fraction, 14
Self Replication Threshold, 14
Site Agent List, 16
Site Editor, 15
Site File Name, 15
Site Name, 15
Species, 12
Species Levels Graph, 6, 12
Species Summary, 19
Stack, 19
Stack Editing, 6, 11
System Requirements, 3

Tags, 16
Tar, 4
Taxation, 14, 16
Text Windows, 8, 17, 19
Trading Excess, 14
Trading Fraction, 13
Trading Resources, 16

Uncompress, 4
UNIX Commands, 2
UNIX Environment Variables, 3
UNIX Shell Prompt, 3
Uptake Masks, 16
Using Ftp, 4, 21

Variant Levels Graph, 7, 12
Verbose Level, 7, 10, 19

Widgets, 3
Window Managers, 5
World Editor, 13
World File Name, 13
World Geography, 13
World Name, 13
World Size, 13
Write Permissions, 18

X Resources, 5, 23
X Windows, 3, 17, 23