Evolution of Differentiation in Multithreaded Digital Organisms

Thomas S. Ray and **Joseph F. Hart**

ATR Human Information Processing Research Laboratories
2-2 Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-02, Japan
ray@hip.atr.co.jp jhart@hip.atr.co.jp
http://www.hip.atr.co.jp/~ray

Abstract

Examination of code execution patterns in different threads of multithreaded digital organisms reveals the clustering of threads into tissues which differ in patterns of code execution. Evolution in a network environment has led to the differentiation of new tissues, which through a division of labor, work together to accomplish the sensory function of a single tissue type of the ancestral organism.

Introduction

(Ray and Hart, 1998) stated:

"The central objective of this project is to study the conditions under which evolution by natural selection leads to an increase in complexity of the replicators. For the purpose of this study, the primary quantitative measure of complexity is the level of differentiation of the multicellular organism. The study begins with the most primitive level of differentiation: two cell types. There are two milestones in the study:

1. The differentiated state persists through prolonged periods of evolution.
2. The number of cell types increases through evolution."

"In the work reported here, only the first of these two milestones has been achieved. There has been no sign of an increase in the number of cell types."

The current study reports the first evidence of an increase in the number of cell types. This evolutionary increase in the level of differentiation of the multicellular digital organisms was detected through the development of new tissue analysis tools.

The digital organisms of this study are self replicating multithreaded machine code organisms living on a network of computers. This system is generally known as Tierra (Ray, T. S. 1991.; Ray, T. S. 1994a.; Ray, T. S. 1994b.; Ray, T. S. 1995.). This study does not address the transition from single threaded to multithreaded organisms as addressed by previous work (Ofria, Adami, Collier and Hughes, 1999.; Ofria, Adami, 1999.), but rather evolution from an ancestral multithreaded state. The intention here is to observe thread differentiation as a function of time rather than environmental complexity, which is addressed by (Ofria, Adami, Collier and Hughes, 1999.).

The digital organisms are able to gather information about other machines on the network, process that information, and use it in making decisions about movements between machines. The digital organisms move between machines in order to improve their access to resources, primarily CPU time, which is their energy resource.

Analogies

Here we are making analogies between some features of digital organisms and organic organisms. The objective of making these analogies is not to create a digital model of organic life, but rather to use organic life as a source of ideas on how to create a richer evolutionary process in the digital medium.

In organic organisms, the "genome" is the complete DNA sequence, of which a copy is found in each "cell". Each cell is a membrane bound compartment, and requires its own copy of the DNA, as the genetic information is not shared across the cell membranes. The entire genome includes many "genes", which are segments of DNA that code for specific functions such as individual proteins.

While each cell contains a complete copy of the genome, each individual cell expresses only a small subset of the genes in the entire genome. The specific subset of genes that are expressed in a cell determine the "cell type". Groups of cells of the same type form a "tissue". Different tissues are composed of cells that have "differentiated" in the sense that they express different subsets of the genes in the genome.

In the digital organisms of this study, the genome consists of the complete sequence of executable machine code of the self replicating computer program. Each thread of a multithreaded process is associated with its own virtual CPU. These threads (CPUs) are considered analogous to the cells. However, the threads of a process all share a single copy of the genome, because they operate in a shared memory environment where the genetic information can easily be shared between CPUs. Duplication of the genome for each thread would be redundant, wasteful and unnecessary. In this detail, our digital system differs quite significantly from the organic system.

The genome of the digital organism includes several segments of machine code with identifiable functions, which are coherent algorithms or subroutines of the overall program represented by the entire genome. These individual algorithms can be considered analogous to the genes. Each thread (CPU) has access to the entire genome, yet each thread will execute only a subset of the complete set of genes in the genome. The specific subset of genes executed by a single thread determine its cell type. Groups of threads of the same cell type form a tissue. Different tissues are composed of threads that are differentiated in the sense that they execute different subsets of the algorithms (genes) in the genome.

Another difference between the organic and digital systems is that in the digital system there is no spatial or geometric relationship between cells. However, there exist logical relationships between threads. For example, the odd threads of a tissue may behave differently from even threads.

Developmental Behavior of the Network Ancestor

The work reported here is focused on the evolution of the differentiated multicellular condition. The multithreaded digital organisms live in a networked environment where spatial and temporal heterogeneity of computational resources (most importantly CPU time) provides selective pressure to maintain a sensory system that can obtain data on conditions on various machines on the network, process the data, and make decisions about where to move within the network.

The experiment begins with a multithreaded ancestral seed program (0960aad) that is already differentiated into three cell types: a sensory tissue, a reproductive tissue, and a copy tissue. The entire seed program includes about 320 bytes of executable machine code. However, no single thread executes all of this code, just as no cell in the human body expresses all of the genes in the human genome.

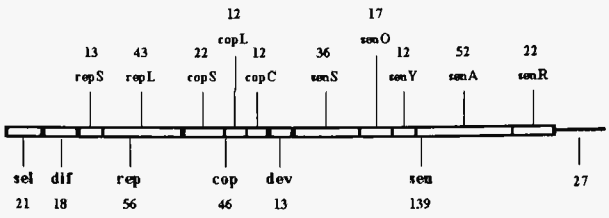

Figure 1: Lower labels indicate the six major genes and their sizes in bytes. Upper labels indicate subdivisions of the major genes, and their sizes.

The network ancestor genome has been somewhat arbitrarily labeled as composed of six genes, some of which have been further subdivided (Figure 1). Two of the genes are executed only during the development from the unicellular to the mature ten celled form (**sel**, **dif**). One gene is executed only by the reproductive tissue (**rep**), and one gene is executed only by the sensory tissue (**sen**). Two genes are executed by the sensory, reproductive, and copy tissues (**cop**, **dev**).

Figure 2 illustrates early development in the ancestor. The ancestor is born as a single undifferentiated thread which begins by executing the **sel** gene to perform the self examination. Execution then flows into the **dif** gene which causes the embryonic cell to split into two cells, which then differentiate into the sensory and reproductive tissues.

The mechanism of differentiation is a conditional jump. When a thread splits into two threads, the value of the dx register,is altered such that the two daughter threads have different values (The value in the dx register is copied from the mother thread to the daughter thread, then both values are shifted left. The value in the daughter thread is then incremented by one). By making conditional jumps based on tests of the values of the dx registers, one thread begins executing the **sen** gene, while the other thread begins executing the **rep** gene.

There exists a mechanism of gene promotion, in which genes promote the expression of other genes, through function calls. In a function call, execution jumps from the location of a call, to the code of the called function (another gene), and when the function is complete, execution returns to the instruction following the location of the call. The **rep** and **sen** genes promote the expression of the **cop** gene, which in turn prompts the expression of the **dev** gene. The **dev** gene is also directly promoted by the **sen** gene (Figure 3).

The function of the **dev** gene is to split a single thread into a tissue of 2^n threads, without differentia-

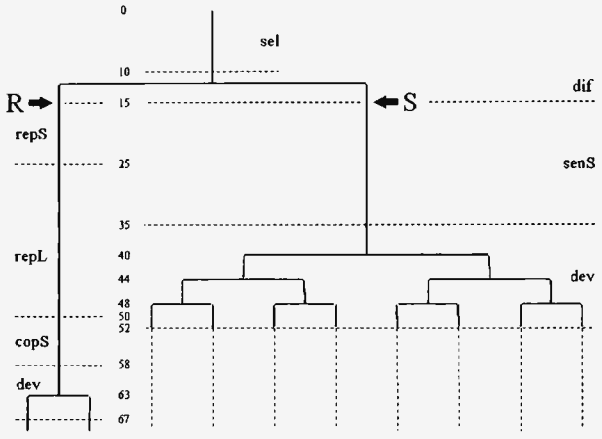

Figure 2: Early development and gene expression in the ancestor, 0960aad. The column of numbers indicates time, in parallel instructions executed by the organism. Horizontal dashed lines show the points of transition between expression of different genes in the threads crossed by the dashed lines. The labels (i.e., **sel**, **dif**, **senS**, etc.) indicate the gene expressed by a thread in a specific time interval. R arrow and S arrow show the point of differentiation into the sensory and reproductive threads.

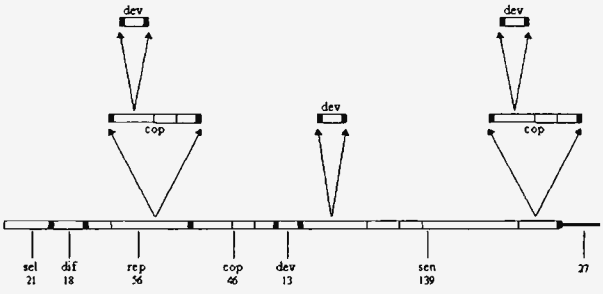

Figure 3: Genes can promote the expression of other genes by making function calls.

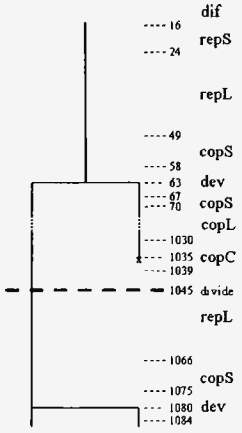

Figure 4: The single reproductive thread calls the **cop** gene to effect the replication of the genome with two threads, after which the daughter is born and sent to another machine on the network. The other machine is chosen based on data processed by the sensory system. The reproductive process repeats until the death of the organism.

tion. The value of n is determined by data in a CPU register.

The **cop** gene performs a string copy function, copying a block of memory from one location to another. This is the means by which the genetic information is replicated from mother to daughter. It is also the means by which sensory data is copied during sensory processing. The **cop** gene parallelizes the string copy function, by calling the **dev** gene and then dividing the data among the 2^n threads. After the data has been copied, all but the original thread halt.

The **cop** gene has three subgenes. The **copS** subgene (copy tissue setup), sets up register values and calls the **dev** gene to parallelize the function. The **copL** subgene (copy loop) copies the data. The **copC** subgene (copy tissue cleanup) halts all but the original copy thread and restores register values before returning from the copy function.

The reproductive thread expresses the reproductive gene, **rep**, which contains two subgenes (Figure 4). The **repS** subgene (reproduction setup) prepares some register values. The **repL** subgene (reproduction loop), is an infinite loop that calls the **cop** gene, spawns a daughter, and repeats the reproduction process. The **cop** gene parallelizes the reproductive tissue into two threads, each of which copies half of the genome.

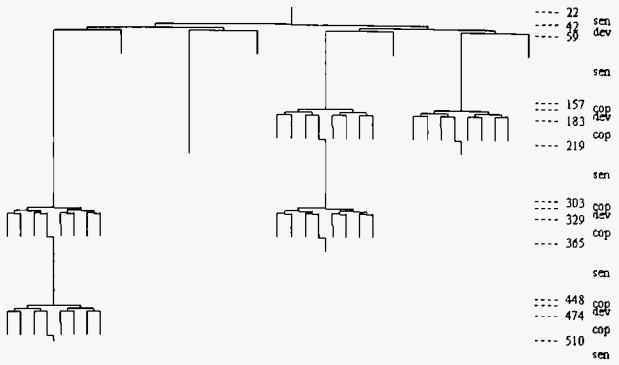

Figure 5: Sensory processing in the ancestor (0960aad).

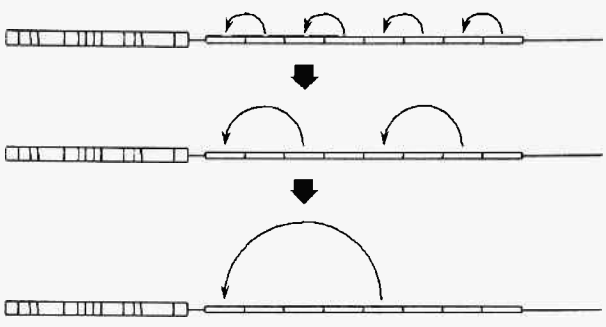

Figure 6: Sensory processing in the ancestor, 0960aad.

In each reproductive cycle after the genome has been copied, the **cop** gene returns control to the **rep** gene, and the reproductive thread executes the **divide** instruction, which spawns the daughter as an independent process. Before giving birth to the daughter, the reproductive thread reads data that has been processed by the sensory system. By this time, the sensory system has examined data from fifteen machines on the network, and placed the IP address of the "best" looking machine at a certain location. At birth the reproductive system sends the daughter to the machine whose IP address is stored at that location. Meanwhile the sensory thread expresses the **sen** gene, beginning with **senS**, which calls the **dev** gene causing the sensory thread to divide into a tissue of eight sensory threads. The **dev** gene then returns control to **sen** and the eight sensory threads begin the complex process of gathering and processing sensory data (Figure 5).

Adjacent to the 320 byte genome is a 512 byte data area used by the sensory system. Each of the eight sensory threads reads a 64 byte TPing data structure into a block of the data area. This initial gathering of sensory information is followed by a series of three pairwise comparisons, which result in the "best" looking data being placed in the leftmost block. It is from this location that the reproductive tissue will find the IP address of the machine to which the daughter will be sent (Figure 6).

After the eight parallel sensory threads have each gathered their TPing data, half of the threads (the odd threads) halt, leaving the four remaining threads to perform the first round of four pairwise comparisons. If the data on the right is "better" than the data on the left, the sensory thread calls the **cop** gene to copy the sensory data from the right block to the left block. The **cop** gene calls the **dev** gene to parallelize the copy process into eight threads. After the sensory data has been copied, seven of the eight copy thread halt before returning control to the **sen** gene.

Two of the four sensory threads then halt, and the remaining two threads perform the second round of two pairwise comparisons. One of the two remaining sensory threads then halts, and there is a third and final comparison leaving the best data in the leftmost position. Note that the **sen** gene only calls the **cop** gene (and thus **dev** gene) if the comparison indicates that the data needs to be copied from right to left. For this reason the pattern of gene expression in the sensory tissue is dependent on the data gathered from the environment.

The **sen** gene is divided into five subgenes. The **senS** subgene (sensory tissue) setup, sets up some register values, calls the **dev** gene to split the sensory thread into a tissue of eight threads, and collects the TPing data. The **senO** subgene (sensory processing coordination), helps the different sensory threads to coordinate the division of the data among themselves, and recognizes the completion of the sensory cycle. The **senY** subgene (sensory system synchronization) causes the odd threads to halt after gathering the TPing data, and forces synchronization of the threads (threads may lose their synchronization because some threads must copy TPing data while others may not). The **senA** subgene (sensory data analysis) compares two sets of TPing data, and decides if it is necessary to copy the data from left to right. The **senR** subgene (sensory data report) calls the **cop** gene if necessary to copy the TPing data from left to right.

Tissue Analysis Methods

The rationale for the current experiment was originally presented by Ray (Ray, T. S. 1995.). Technical details of the implementation have been reported in Charrel (Charrel, Agnes, 1995.) and Ray (Ray, T. S.

1997.; Ray, T. S. 1998.). Further details are available on the web at:

http://www.hip.atr.co.jp/~ray/tierra/
netreport/netreport.html.

Thus only those experimental methods new to the current report will be presented here. These new methods are those involved in the analysis of tissue types.

If new tissues do in fact evolve in Tierra, they may differentiate gradually, starting with only slight differences which then gradually increase through evolution. To help us get a clear and realistic picture of this process, we must classify threads into tissues through a careful multistep step process. This process will be described through the example of the analysis of the ancestral organism (0960aad).

In Tierra, a tissue can be strictly and objectively defined as the set of threads which execute exactly the same code. Any two threads which execute a different set of Tierran machine code instructions will be classified into different tissues. We will refer to tissues defined in this manner as "strict tissues".

However, this strict definition is only a starting point for the grouping of threads into tissues. There may exist slight difference in the code executed by two threads that will be of little significance. For example, when a single thread divides into a multi-thread tissue, the last threads to be formed generally execute slightly fewer instructions than the threads which formed earlier. We can recognize groups of trivially different tissues by doing a cluster analysis on the code expression patterns of the strict tissues. Strict tissues that are only trivially different should be grouped together as a single tissue type.

Even two threads which execute exactly the same algorithm may execute different code due to conditional branches or halts, based on variable data gathered from the environment. Due to such conditional behavior, the same thread could execute different code at different times, depending on the data that it gathers from the environment. We will refer to differences in code execution due to conditional behavior as "behavioral differences". Strict tissues which are behaviorally different can be recognized only through a careful examination of their algorithms. Strict tissues that are behaviorally different should be grouped together as a single tissue type.

The classification of threads into strict tissues is an automated process, based on data gathered during a finite period of observation. Termination of the period of observation generally causes some threads to terminate before fully completing their algorithm. These incomplete threads are spuriously classified as distinct tissues. Tissues whose distinction is due to such "observational differences" should be grouped with tissues which have completed the same algorithm.

When two or more strict tissues are grouped together as a single tissue because they are trivially, behaviorally, or observationally different, we call the grouping a "narrow tissue". The common theme in grouping strict tissues into narrow tissues is to recognize the unity of threads which execute the same algorithm.

The definition of narrow tissues on the basis of algorithms can also lead to the need to split a single strict tissue into two narrow tissues. This situation arises uniquely in the case of undifferentiated embryonic threads which later differentiate. These threads first execute the algorithm for embryonic development, and after differentiation execute the algorithm of a mature tissue, such as the reproductive tissue.

Narrow tissues can be further grouped into "broad tissues", based on their functional actions. The broad tissues recognized in this study are embryonic, copy, reproductive, and sensory. This study will focus on the evolution of new narrow tissue types in the sensory broad tissue.

The first step in the tissue analysis process is to group threads into strict tissues on the basis of their patterns of code execution, a process that can be completely automated by recording the code executed by each thread of sample organisms living in a network environment. This leads to the groupings shown in Table 1.

The execution patterns of the ten strict tissues are then compared in a similarity matrix (Table 2) so that we can perform a cluster analysis.

Because the amount of code executed by two tissues may be different, the similarity may not be symmetric. For example, tissues 0 and 6 show a nearly symmetric relationship, with a 5% and 6% similarity. Tissues 7 and 9 on the other hand are highly asymmetric, with 100% and 12% similarities (meaning that tissue 7 executes all the code of tissue 9, while tissue 9 executes only 12% of the code of tissue 7).

The similarity matrix is then subjected to a cluster analysis using the PC ORD 4 software from MjM Software (http://www.ptinet.net/~mjm/pcordwin.htm). For the ancestral organism, 0960aad, it produced the cluster diagram shown in Figure 7.

A glance at the cluster diagram suggests some natural groupings, however, these must be considered in the light of a careful analysis of the functions of the

strict	# threads	sel	dif	repS	repL	copS	copL	copC	dev	senS	senO	senY	senA	senR
0	1	sel	dif	repS	repL	copS	copL	copC	dev					
1	1		dif			copS	copL	copC	dev	senS	senO	senY	senA	senR
2	17					copS	copL	copC	dev	senS	senO	senY	senA	senR
7	3					copS	copL		dev	senS	senO	senY	senA	senR
3	28								dev	senS	senO	senY		
6	1								dev	senS	senO	senY	senA	
4	155					copS	copL	copC	dev					
5	114					copS	copL	copC	dev					
8	9					copS	copL		dev					
9	12					copS	copL		dev					

Table 1: A listing of the ten strict tissues identified in the ancestor, 0960aad, the number of threads found in each tissue, and the genes expressed in each tissue.

	T0	T1	T2	T3	T4	T5	T6	T7	T8	T9
T0	1.00	0.43	0.39	0.04	0.21	0.22	0.05	0.29	0.17	0.16
T1	0.24	1.00	0.81	0.20	0.12	0.12	0.52	0.74	0.10	0.09
T2	0.27	0.99	1.00	0.26	0.14	0.15	0.64	0.91	0.12	0.11
T3	0.11	0.94	1.00	1.00	0.11	0.11	1.00	0.97	0.11	0.11
T4	0.95	0.95	0.95	0.19	1.00	1.00	0.19	0.71	0.71	0.71
T5	0.95	0.95	0.95	0.18	0.95	1.00	0.23	0.73	0.73	0.68
T6	0.06	0.98	0.99	0.40	0.04	0.06	1.00	0.98	0.06	0.04
T7	0.21	0.99	1.00	0.28	0.12	0.13	0.70	1.00	0.13	0.12
T8	1.00	1.00	1.00	0.25	0.94	1.00	0.31	1.00	1.00	0.94
T9	1.00	1.00	1.00	0.27	1.00	1.00	0.27	1.00	1.00	1.00

Table 2: Strict tissue similarity matrix for the ancestor 0960aad. Table entries are computed by dividing the number of instructions executed in common by the two tissues by the total number of instructions executed by the tissue of the row.

Broad	Narrow	Strict	Function
Embr/Repr	A	0	embry/reprod
Copy	B	4, 5, 8, 9	copy
Sensory	C	1, 2, 3, 6, 7	full sensory

Table 3: Grouping of strict tissues into narrow tissues.

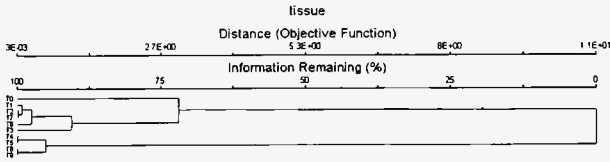

Figure 7: cluster diagram - 0960aad

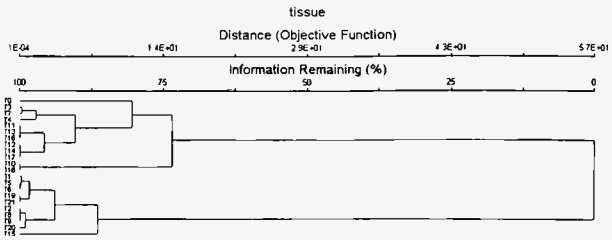

Figure 8: cluster diagram - 3414aaj

different strict tissues, and the role of trivial, behavioral, and observational differences. These considerations lead to the groupings given in Table 3.

Narrow tissue A consists of the original embryonic thread, which eventually differentiates into a reproductive thread. This one thread strict tissue is unique by virtue of being the only thread to execute the embryonic code (**sel**, **dif**). However, it also expresses the **rep** gene and the **cop** gene.

Narrow tissue B expresses the copy gene (**cop**), which is promoted by the reproductive code to copy the genome, and by the sensory code to copy TPing sensory data. The threads of tissue B are created by the **dev** gene for the parallelization of the copy function, and halt when the data has been copied, thus they express only the **dev** and **cop** genes.

Narrow tissue C expresses the sensory gene, **sen**. The algorithm of the sensory gene includes a data reduction loop in which the odd threads halt, the even threads compare the data on the left and right, and depending on the results of the comparison, may or may not promote the copy gene to copy the TPing data.

Within this sensory gene we find two kinds of behavioral difference: tissue 6 does not copy the data while tissues 1, 2, and 7 do copy the data; tissue 3 differs from tissues 1, 2, 6, 7 in that it halts early in the first sensory cycle. We also find observational differences: tissue 7 does not finish the copy loop during the observation period while tissues 1 and 2 do finish. We also find trivial differences: tissue 1 is the original sensory thread, originating in the **diff** gene, and thus differs slightly from the rest of the sensory tissues in that it executes a small part of the **diff** gene before entering the sensory code.

Developmental Behavior of an Evolved Organism

The tissue analysis procedure was applied to an organism captured after five days of evolution (3414aaj). An examination of homology between the genome of 3414aaj and the genome of the ancestor revealed the existence of all thirteen ancestral genes in the evolved genome. These thirteen genes accounted for the entire evolved genome; there were no new genes added or lost.

Classification of the threads of 3414aaj into strict tissues produced the pattern shown in Tables 5. The cluster analysis (Figure 8) together with consideration of trivial, behavioral, and observational differences leads to the classification into narrow and broad tissues shown in Table 4.

The grouping of strict tissues into narrow tissues in 3414aaj follows the same procedures detailed for the ancestor in the methods section above. However, in the end we are left with two narrow sensory tissues which differ both in their code execution pattern and their function.

Narrow tissue D loads the TPing data into the sensory data buffers using the **senO** gene. Narrow tissue E does not execute **senO**, but does perform the comparison of the left and right data using the **senA** gene, and uses the **senR** gene to call the **cop** gene if it is necessary to copy the data.

After loading the TPing data buffers, the threads of narrow tissue D go on to execute the **senA**, **senR**, and possibly the **cop** gene. The comparisons and copies made by tissue D have no effect, however, because the same data is later operated on by the threads of tissue E.

It should be noted that tissue E is a sub-set of tissue D, in the sense that tissue D executes all the code of tissue E, while tissue E executes only a portion of the code of tissue D. Where the execution of code overlaps between the two, in the genes **senA**, **senR**, **copS**, and **copL**, the code has no effect in tissue D. The functional division of labor between the two tissues is thuse absolute, even where there is overlap of code execution.

Broad	Narrow	Strict	Function
Embr/Repr	A	0	embry/reprod
Reproductive	B	3,4,7,12,14,17	reprod
Copy	C	10,11,13,16,18	copy
Sensory	D	1,2,5,6,8,9	sensory load
Sensory	E	19,20,21	compare, cpy
Vestigial	F	15	vestigial

Table 4: Grouping of strict tissues into narrow and broad tissues in a five day evolved organism, 3414aaj.

strict	#T	sel	dif	repS	repL	copS	copL	copC	dev	senS	senO	senY	senA	senR
0	1	sel	dif	repS	repL	copS	copL		dev					
3	1				repL	copS	copL	copC	dev					
4	1				repL	copS	copL		dev					
7	1				repL	copS	copL	copC	dev					
12	2				repL	copS	copL	copC	dev					
14	5				repL	copS	copL	copC	dev					
17	8				repL	copS	copL	copC	dev					
11	2					copS	copL	copC	dev					
13	5					copS	copL	copC	dev					
16	8					copS	copL	copC	dev					
10	38					copS	copL		dev					
18	48					copS	copL		dev					
1	1		dif			*copS*	*copL*		dev	senS	senO	*senY*	senA	*senR*
2	1					copS	copL		dev	senS	senO	*senY*	senA	senR
5	2					copS	copL		dev	senS	senO	*senY*	senA	senR
6	1					*copS*	*CopL*		dev	senS	senO	*senY*	senA	*senR*
8	2								dev	senS	senO	*senY*	senA	
9	1								*dev*	*senS*	*senO*	*senY*	*senA*	
19	3					copS	copL		dev				senA	senR
20	4					copS	copL		dev				senA	senR
21	1					*copS*	*copL*		*dev*				*senA*	*senR*
15	8										*senO*	*senY*		

Table 5: A listing of the twenty-two strict tissues identified in the evolved organism, 3414aaj, the number of threads found in each tissue, and the genes expressed in each tissue. Genes named in italics are expressed in the tissue, but have no effect (generally because another thread overwrites the same data area).

Discussion

The ancestor (0960aad) has a single sensory tissue type which performs all of the sensory functions. In the organism resulting from five days of evolution, 3414aaj, these sensory functions have been divided into two separate tissue types, with a functional division of labor. The threads of tissue D load the TPing data into buffers, and the threads of tissue E compare the data and copy from left to right if necessary.

While the functional division of labor is complete and clear cut, the difference in code execution patterns is not as clean. Tissue D also executes the code executed by tissue E, but in tissue D that common code has no effect.

The data presented here reflect only our first observations of the evolution of higher levels of differentiation in multithreaded digital organisms. We expect that further observations are likely to reveal additional examples, and possibly examples of yet greater and more clear cut differentiation.

References

Charrel, Agnes. 1995. Tierra network version. ATR Technical Report TR-H-145. http://www.hip.atr.co.jp/~ray/pubs/charrel/charrel.pdf

C. Ofria, C. Adami, T.C. Collier, and G.K. Hsu. 1999 Evolution of Differentiated Expression Patterns in Digital Organisms, Lect. Notes Artif. Intell. 1674, 129-138 Proceedings ECAL'99 http://www.krl.caltech.edu/avida

C. Ofria, C. Adami. 1999 Evolution of Genetic Organization in Digital Organisms Proc. of DIMACS workshop on Evolution as Computation, Jan 11-12, Princeton University, L. Landweber and E. Winfree, eds. Springer Verlag, p. 167 http://www.krl.caltech.edu/avida

Ray, T. S. 1991. An approach to the synthesis of life. In: Langton, C., C. Taylor, J. D. Farmer, and S. Rasmussen eds., *Artificial Life II, Santa Fe Institute Studies in the Sciences of Complexity*, vol. XI, 371-408. Redwood City, CA: Addison-Wesley. http://www.hip.atr.co.jp/~ray/pubs/tierra/tierrahtml.html

Ray, T. S. 1994a. Evolution, complexity, entropy, and artificial reality. *Physica D* 75: 239-263. http://www.hip.atr.co.jp/~ray/pubs/oji/ojihtml.html

Ray, T. S. 1994b. An evolutionary approach to synthetic biology: Zen and the art of creating life. *Artificial Life* 1(1/2): 195-226. http://www.hip.atr.co.jp/~ray/pubs/zen/zenhtml.html

Ray, T. S. 1995. A proposal to create a network-wide biodiversity reserve for digital organisms. ATR Technical Report TR-H-133. http://www.hip.atr.co.jp/~ray/pubs/reserves/reserves.html

Ray, T. S. 1997. Selecting Naturally for Differentiation. In: Koza, John R., Kalyanmoy Deb, Marco Dorigo, David B. Fogel, Max Garzon, Hitoshi Iba, and Rick L. Riolo eds. *Genetic Programming 1997*: Proceedings of the Second Annual Conference, July 13–16, 1997, Stanford University, 414-419. San Francisco, CA: Morgan Kaufmann. http://www.hip.atr.co.jp/~ray/pubs/gp97/gp97.html

Ray, T. S. 1998. Selecting Naturally for Differentiation: preliminary evolutionary results. *Complexity*, 3(5): 25-33. John Wiley and Sons, Inc.

Ray, T. S. and Joseph Hart. 1998. Evolution of Differentiated Multithreaded Digital Organisms. In: *Artificial Life VI* proceedings, C. Adami, R. K. Belew, H. Kitano, and C. E. Taylor eds., 295-304. The MIT Press, Cambridge.