

Internal Constraints and Ecology in Evolution: A Case Study in Tierra

Chrystopher L. Nehaniv

Adaptive Systems Research Group
Faculty of Engineering and Information Sciences
University of Hertfordshire
College Lane
Hatfield, Herts AL10 9AB
United Kingdom

C.L.Nehaniv@herts.ac.uk

<http://homepages.feis.herts.ac.uk/~nehaniv>

Abstract

We study the evolution of self-replicating assembly language computer programs in T. S. Ray's Tierra system, using a variant mechanism for the replicators to determine their own size. A new ancestor is introduced which uses inherited code for carrying arithmetic operations that give its correct size rather than self-measuring (as in the original Tierra studies).

The courses of evolution from Ray's ancestor and our new ancestor are compared. The mechanism for size determination used by replicators proves to be remarkably conserved. Largely similar ecologies evolve in both cases. Parasites, hyperparasites, recurring immunity to parasites and its circumvention are all observed in evolution from either ancestor. But niches for social replicators and social cheating do not appear to arise (apparently due to lack of necessity of end templating in replicators). However a new type of metabolic informational parasitism occurring in evolution from the new ancestor is characterized and exemplified. This new opportunity creates a set of new parasite and hyperparasite niches.

Self-measurement vs. self-description for a replicator to determine its own size proved to be an inviolable internal constraint of evolution in these studies. In turn, this affects some aspects of variability (e.g. of size and also in the use of end-marker templating). Altering the size determination mechanism leaves much of the ecological dynamics unchanged, but has an extreme impact on the potential for certain types of niche creation and subsequent ecological dynamics.

Finally the smallest known Tierrans using each of these two strategies (self-measuring and genetic encoding of size calculation) are given explicitly.

1 Introduction

The question is often asked whether and how organic evolution would have been qualitatively the same if it were restarted from the beginning, e.g. (Fontana & Buss 1994). That is, what would be conserved and what would be different in different "runs" of evolution starting from similar initial conditions? Evolution in digital media that takes place on human timescales allows us to answer this question experimentally for model systems. In this work, we address this problem experimentally by creating a self-replicating entity to run under Thomas Ray's Darwinian operating system,

Tierra (Ray 1992), in which self-replicating assembly language programs are subject to the conditions affording evolution (including limited resources, heritability of fitness, and conditions that result in variability but not extreme brittleness in the face of it).

We introduce a new ancestral creature that uses a method of self-replication that differs in a small but essential manner from that of Ray's original work. In particular, Ray's ancestral Tierran measures itself, acquires space equal to its own size, and copies itself into that space. Self-inspection in reproduction has been studied in theoretical biology to some extent (Laing 1977), and is used by Ray's ancestor in the measuring and copying portions of its reproductive cycle. Our variant ancestor is the same except that its genome encodes instructions for determining the creature's size instead of encoding instructions for self-measurement. Thus, in the new variant's case, measuring of the self becomes unnecessary as size information is heritable. Nevertheless, the two ancestors are of the same size and of similar efficiency since the genetic encoding of size requires about the same amount of the genetic material as the genetic information coding for self-measurement, and the time required to execute both is similar.

Tierra source code for the new ancestral creature is given with explanatory annotations in Appendix A. For the original results with Tierra using Ray's size 80 ancestor, as well as for details of the Darwinian operating system, Tierran CPUs and assembly language, the reader may consult (Ray 1992). The study here uses the same version of the Tierra language as in that seminal paper.

2 Evolutionary Dynamics and Ecologies Compared

We found that, with initial conditions modified in this way, once again a diverse ecology of Tierran entities evolves including parasites, hyperparasites, and other niches, which are qualitatively similar in the kinds and scope of ecological relationships to those observed in Ray's work (1992). *Parasites*, in the sense used here and (Ray 1992), cannot replicate on their own but require resources from others in order to replicate. Informational parasites read the code of a host — typically for copying their own instructions into a daughter in the classical work (Ray 1992) — but do not take control of its CPU and arise almost immediately in both models.

Some differences do occur in the type of niches that become available depending on whether *self-measurement* (as in Ray's ancestor) or *self-description* (as in the new ancestor) is used. Obligatory social replicators and social hyper-parasites were not observed in evolutionary runs starting with the new ancestor. However, niches allowing a new type of metabolic informational parasitism became available to descendants of the new ancestor.

2.1 Internal Constraints and Ecological Consequences

2.1.1 Constraints on Variability in Size

Despite striking similarities, certain interesting constraints on evolved creature sizes are observed in the course of evolution from the variant ancestor. These are related to ease with which various sizes may be encoded in the genome using the arithmetic operations in the language of Tierran genomes and the variational properties of mutational effects on such encodings.

Change in the size of descendants of Ray's ancestor did not involve special properties of arithmetic calculation since subtraction of starting address from ending address allowed a replicator to determine its own size. For this mechanism, no special constraints need be imposed on the size.

In contrast, size calculation using our new ancestor was done by simple assembly arithmetic: doubling (shifting), decrementing or incrementing registers, or flipping their lowest order bit. Thus beginning with the size 80 ancestor, most commonly parasites of size 40 appear next (if one doubling operation is lost). Changes in size by plus or minus one in the course of calculation or by powers of two are the most common. Thus variability of change in size is constrained by ease of ability to calculate new variant sizes.

2.2 Constraints on Self-Examination vs. Self-Description

Moreover, we note that observed longterm evolution (four separate runs of over 10 billion Tierra instruction executions as compared with published data on evolution from the original ancestor) from these two ancestors differs and is remarkably conservative in the following respects:

1. Successful descendents of Ray's ancestor in evolutionary runs studied all employ a self-measuring step in self-replication.
2. Successful descendents of our variant ancestor all use inherited code for arithmetic operations to calculate their own sizes.

Allowing evolution to continue (much beyond the time scale of Ray's original experiments) in both scenarios showed quantitatively very similar reduction in ultimate genome sizes in both cases (sizes down to 36 or 24 were commonly achieved). Nevertheless the mode of determining size (self-measurement or self-description) always persisted without exception throughout the course of evolution.

2.3 Metabolic Informational Parasitism: Opportunities for New Ecological Niches

A type of informational parasitism possible with the new ancestor gives rise to a niche for parasites that was unavailable in Ray's original work. Since the self-measuring mechanism requires locating an end template, replicators using this method need to have an end template (or, in the case of sociality, be adjacent to a creature that completes the end template.) Similarly, sociality that uses the end template of a neighbor as part of the start template of a replicator is not supported if there is no end template in the neighbor. With the genetic encoding of size, this constraint falls away.

Now, since matching an end template is not necessary, it becomes possible to call the code of other entities in order to calculate one's own size. Parasites evolved from Ray's ancestor needed to measure themselves *before* calling or jumping to a host's copy loop. A parasite evolved from the new ancestor needs to arrange that subsequently executed code should (1) calculate its correct starting address (from contents of its registers *ax* and *cx*), (2) calculate its size, (3) allocate a space for an offspring of appropriate size, and (4) copy the genetic material of the parasite to the space allocated to this offspring. Only (3) and (4) are available as informational parasitic options given the self-measuring mechanism. In evolutionary runs from the new ancestor, parasites of this type were observed, but also parasites capable of using host code to carrying out all of (1)-(4) evolved.

Parasites using host code only for (3) and (4) quickly evolve from the new or original ancestor. Eventually parasites also using host code for (1) and (2) appear, but only in evolution from the new ancestor.¹

¹Reproductive function (2), but not (1), might also be possible for parasites descended from Ray's ancestor to exploit in their hosts since it only involves a simple subtraction of registers (given by the single instruction `subAAC`).

The new type of parasitism is a *metabolic parasitism*, in that the host's machinery to calculate its own size is high-jacked to produce a new product, calculation of the size of the parasite. It is nevertheless an informational parasitism as the information encoded in the genome of the host (but not the processing power) in the host is exploited. We therefore classify it as a form of *metabolic informational parasitism*.

For a concrete example of a metabolic informational parasite in the course of evolution using the new ancestor, in one case after an initial informational parasitism is wiped-out by immunity, there was a long stable period of size 80 replicators which were resistant to parasitism. Later, 63 and 61 were the dominant large sizes of replicators. These calculated their sizes as follows: Register ax contains the host's address plus a offset equal to the start template size ($ax = @ + cx$, where $@$ denotes the start address of the host, and the contents of register $cx = 4$ encode template size). The code for calculating the host's own size replaces the value cx by $16 \times cx - 3 = 61$, the host size. A parasite with start template of size $cx = 1$ calls the host at an early stage, it executes the same code but this time yielding a value in register cx of $16 \times cx - 3 = 13$, the parasite size. The parasite then executes all subsequent code of the host to produce numerous offspring. Similarly, hosts of size 63 had parasites of corresponding size 15, using host code to calculate their size as $16 \times cx - 1$.

A sequence of 8 instructions at the beginning of a parasite was sufficient for this type of parasitism, and was actually insensitive to the actual size of the parasite.² Thus parasites with various terminations beyond the initial instructions formed a kind of 'quasi-species' where the only important heritable genetic information was contained in this initial part of the creature.³

In a separate independent evolutionary run, hosts calculated their own size as 36 starting with its start template size in cx of 3. A parasite uses the same code to calculate its own size as 20 starting from its template size $cx = 2$. This example is detailed in Appendix B.

Just as for parasites only using host code for reproductive functions (3) and (4) mentioned above, *hyperparasites*, which reset the registers of a calling parasitic CPU to values for replicating the host arise in both models. Subsequently, immunity to such parasites and also hyperparasitism wherein the host resets the parasite's CPU to copy the host arise. But from time to time periodic outbreaks of parasitism recur.

2.4 Remarks Concerning Social Replicators

Unlike the results of Ray (1992), we did not find evidence of stable social replicators. The reasons for this difference are discussed here. Due to the requirement of an end template for using the self-measuring mechanism (discussed above), replicators evolved from Ray's ancestor have templates at their ends as in their ancestor. These provide the possibility to seek an address backwards or forward into an adjacent creature during self-measurement (illustrated on pp. 385-387 of (Ray 1992)). With a high degree of homogeneity, this allowed obligatory social replicators to evolve,

²The initial 8 instructions are `nop0 adrb nop1 jmpo nop1 nop1 nop1 nop0` followed by anything but a `nop`. Here `1 1 1 0` matches the template in the host preceding the code for (1)-(4). The parasite has $cx = 1$ and as long as the size calculated from this exceeds 8, these instructions will be reproduced in an offspring, generally not identical to the parasite. Thus this class of parasites forms a kind of *quasi-species* in which the 'true' replicator is the initial sequence of the parasite and the rest is subject to neutral variation. This type of parasitism is possible since an end template never needs to be matched, as mentioned above.

³Another example of a non-parasitic quasi-species were a variety of individuals of size 38 evolved from the new ancestor. They allocate memory of size 38 for the offspring, but only bother to copy their first 37 instructions to the daughter. The last instruction is never normally executed and thus is subject to neutral variation. This latter type of quasi-species might also arise during evolution from Ray's ancestor.

and then thereafter social hyper-parasite cheaters (which, sandwiching themselves between obligatory social replicators, steal a neighbor's CPUs, for example, by giving their own starting address rather than the host's starting information, or by having a start template that is matched, when the host searches back into its neighbor for a start template).

Creatures using the new genetically encoded mechanism of using arithmetic operations for determining their own size only needed a start template and so cannot in general rely on any template that immediately precedes them in their environment. Therefore this type of sociality seems never to evolve. Although social cheater parasites seem to have been avoided with the new mechanism, it is open to attacks by informational metabolic parasites (discussed above).

3 Conclusions: The Impact of Internal Constraints on Evolution

Despite the remarkable ecological similarities observed when evolution begins with either ancestor, the characteristic method that creatures used to determine their own length is remarkably persistent through time. As such, this feature of the method of replication is a concrete example of a software evolutionary analogue to a "frozen accident" in organic biology. That is, it is a feature which, although it might in no sense be optimal, cannot be varied and it constrains all subsequent evolution. Varying the "accident" in the course of an evolutionary run, either results in numerous cascade effects that affect the ability to survive and reproduce, or, is not feasible given the genotype-phenotype relationship and variability mechanisms.

This observed constraint in Tierra is not an environmental or ecological constraint on the mode of self-reproduction. This is clear since we have an example of evolutionary runs with a similar ecology in which the constraint is violated: Ray's evolutionary runs illustrate that self-measuring can be used instead of inheriting size information. Conversely, our runs show that this is not necessary. Hence the constraint on the mode of determining one's own size is not an external constraint, an ecological constraint, nor a selective constraint. Instead, we have an example of what biologists refer to as *internal constraint* in evolution. (See (Arthur 2000, ch. 2.5) for a recent discussion of different types of constraints on evolution.) An alternative way of understanding this, using Sewall Wright's metaphor of fitness landscapes, is that the two modes are on different fitness peaks and there is no path for evolution to traverse between them without greatly reducing fitness.

Aspects of the early or initial conditions of evolution can persist with inertia carrying them throughout time, as illustrated by absence or presence of self-measuring in Tierra replicators. Other aspects may arise in qualitatively similar ways independent of initial conditions, as illustrated by, size-reduction, similar niche types and most ecological relations in evolution from the two ancestors.

This study also illustrates that internal constraints, such as mode of replicator size determination in Tierra, can have larger evolutionary dynamic and ecological consequences. In the experimental studies examined here, the internal constraint on the method of size determination by self-measurement vs. self-description persisted in all subsequent evolution from an ancestor using one or the other method. *The evidence suggests that this reproductive constraint had repercussions at the much larger level of ecological dynamics: This internal constraint determined*

- (1) *whether social replicators and associated ecological niches may evolve, and*
- (2) *whether certain types of parasitism and associated niches may arise.*

This suggests that the internal constraints in biological evolution may also have far-reaching consequences for ecological evolutionary dynamics.

4 Epilogue: Some Small Tierrans

In addition to a discussion of these ancestors and their descendants, we also display and examine the smallest known self-replicating Tierrans, obtained from small modifications of creatures evolved in this course of this work.

The smallest previously known non-parasitic self-replicating Tierran observed to evolve has size 22. A human-modified version of this of size 21, designed by Manor Askenazi, is social and loses its CPU after replication (according comments on its source code in the Tierra distribution). This creature determines its own size using a self-measuring mechanism.

The smallest known independently (non-parasitic, non-social) self-replicating Tierran 0023c1n which uses self-description, i.e. genetically encoded arithmetic calculation of its own size (23 instructions), is shown in Appendix C.

An even smaller Tierran uses self-measuring (as in Ray's ancestor) and is a human-designed variant of an evolved Tierran. This creature, 0018c1n consists of only 18 instructions and is the smallest known independently self-replicating Tierra (also shown in Appendix C).

These two small Tierrans were designed by the author based on reducing the size of evolved creatures. Inoculating the soup with either creature very soon results in the evolution of slightly larger Tierrans than these ancestors. For example the size 23 ancestor gives rise to size 25 and 24 descendants and becomes extinct. Size 24 then persists, apparently indefinitely (experimentally, at least for 7.5 billion time steps) and always using the arithmetic method of determining its own length. Evolution from the size 18 ancestor, yields sizes in the range 19–23, size 18 becoming extinct, with the greatest number of creatures at size 20, and persists in using taking the difference of start and end template addresses for self-measurement.

References

- Arthur, W. (2000), *The Origin of Animal Body Plans: A Study in Evolutionary Developmental Biology*, paperback edn, Cambridge University Press.
- Fontana, W. & Buss, L. W. (1994), 'What would be conserved if 'the tape were run twice'?', *Proceedings of the National Academy of Sciences, USA* **91**, 757–761.
- Laing, R. (1977), 'Automaton models of reproduction by self-inspection', *Journal of Theoretical Biology* **66**, 437–456.
- Ray, T. S. (1992), An approach to the synthesis of life, in C. G. Langton, C. Taylor, J. D. Farmer & S. Rasmussen, eds, 'Artificial Life II, volume X of SFI Studies in the Sciences of Complexity', Addison-Wesley, pp. 371–408.

Appendix A: New Tierran Ancestor

Tierra source code for the new ancestral creature is given on the next page.

The section comprising instructions 16 to 22 is the critical section that allows the creature to determine its own length. In the new ancestral creature this section encodes genetically that the length of the creature is 80. Register cx contains the number 4 (a template size, used in locating the start of either ancestor, and putting this starting address in register ax at instruction 7). This section manipulates register cx in binary arithmetic to change its value to 80 as follows:

$$4 = (100)_2 \xrightarrow{\text{not}0} 5 = (101)_2 \xrightarrow{\text{sh}1} 10 = (1010)_2 \xrightarrow{\text{sh}1} 20 = (10100)_2 \\ \xrightarrow{\text{sh}1} 40 = (101000)_2 \xrightarrow{\text{sh}1} 80 = (1010000)_2,$$

where `not0` flips the low order bit of cx and `sh1` shifts the register left, doubling the number represented. At the end of this manipulation register cx contains the length of the creature. The value of cx is then used in memory allocation for space for a daughter, and will be decremented every time an instruction is copied from mother to daughter. When register cs is decremented to zero, control is returned from the copying loop, the creature divides (giving the daughter its own CPU), and the reproduction loop begins again.

In the code, Ray's original instructions (R) in the old ancestor are given following their replacements in the new ancestor, where they differ.

Ray's code in the critical section searches forward for the address of the end template and subtracts this from the start of mother's address (which is in register cx) in instructions 21 and 22. Instructions 19 and 20 are unused by the new ancestor, but in the old ancestor, together with instructions 17 and 18, they encode [the complement of] the end of creature template. The reader is referred to (Ray 1992) for a full description of how template matching works in Tierra and for details of the Tierra language and Tierra's Darwinian operating system.

Source code for the Tierrans in these appendices and further information on data gathered in the evolutionary experiments using the new ancestor are available at

<http://homepages.feis.herts.ac.uk/~nehaniv/tierra/>

```

genotype: 0080c1n parent genotype: human
1st_daughter: flags: 0 inst: 827 mov_daught: 80 breed_true: 1
2nd_daughter: flags: 0 inst: 809 mov_daught: 80 breed_true: 1
comments: the new ancestor, knows computes its own size as 80

```

```

nop1 ; 110 01 0 beginning marker
nop1 ; 110 01 1 beginning marker
nop1 ; 110 01 2 beginning marker
nop1 ; 110 01 3 beginning marker
zero ; 110 04 4 put zero in cx
not0 ; 110 02 5 put 1 in first bit of cx
shl ; 110 03 6 shift left cx (cx = 2)
shl ; 110 03 7 shift left cx (cx = 4)
movDC ; 110 18 8 move cx to dx (dx = 4)
adrb ; 110 1c 9 get (backward) address of beginning marker -> ax
nop0 ; 100 00 10 complement to beginning marker
nop0 ; 100 00 11 complement to beginning marker
nop0 ; 100 00 12 complement to beginning marker
nop0 ; 100 00 13 complement to beginning marker
subAAC ; 110 07 14 subtract cx from ax, result in ax
movBA ; 110 19 15 move ax to bx, bx now contains start address of mother
not0 ; 110 02 16 put 1 in first bit of cx (cx = 5)
**** R: adrf ; get (forward) address of end marker -> ax
shl ; 110 03 17 shift left cx (cx = 10)
**** R: nop0 ; complement to end marker
shl ; 110 03 18 shift left cx (cx = 20)
**** R: nop0 ; complement to end marker
nop0 ; 100 00 19 complement to end marker
nop1 ; 100 01 20 complement to end marker
shl ; 110 03 21 shift left cx (cx = 40)
**** R: incA ; increment ax, to include dummy instruction at end
shl ; 110 03 22 shift left cx (cx = 80)
**** R: subCAB ; subtract bx from ax to get size, result in cx
nop1 ; 110 01 23 reproduction loop marker
nop1 ; 110 01 24 reproduction loop marker
nop0 ; 110 00 25 reproduction loop marker
nop1 ; 110 01 26 reproduction loop marker
mal ; 110 1e 27 allocate space (cx) for daughter, address to ax
call ; 110 16 28 call template below (copy procedure)
nop0 ; 100 00 29 copy procedure complement
nop0 ; 100 00 30 copy procedure complement
nop1 ; 100 01 31 copy procedure complement
nop1 ; 100 01 32 copy procedure complement
divide ; 110 1f 33 create independent daughter cell
jmpo ; 110 14 34 jump to template below (reproduction loop)
nop0 ; 100 00 35 reproduction loop complement
nop0 ; 100 00 36 reproduction loop complement
nop1 ; 100 01 37 reproduction loop complement
nop0 ; 100 00 38 reproduction loop complement
ifz ; 000 05 39 dummy instruction to separate templates
nop1 ; 110 01 40 copy procedure template
nop1 ; 110 01 41 copy procedure template
nop0 ; 110 00 42 copy procedure template
nop0 ; 110 00 43 copy procedure template
pushA ; 110 0c 44 push ax onto stack
pushB ; 110 0d 45 push bx onto stack
pushC ; 110 0e 46 push cx onto stack
nop1 ; 110 01 47 copy loop template
nop0 ; 110 00 48 copy loop template
nop1 ; 110 01 49 copy loop template
nop0 ; 110 00 50 copy loop template
movii ; 110 1a 51 move contents of [bx] to [ax] (copy one instruction)
decC ; 110 0a 52 decrement cx (size)
ifz ; 110 05 53 if cx == 0 perform next instruction, otherwise skip it
jmpo ; 110 14 54 jump to template below (copy procedure exit)
nop0 ; 110 00 55 copy procedure exit complement
nop1 ; 110 01 56 copy procedure exit complement
nop0 ; 110 00 57 copy procedure exit complement
nop0 ; 110 00 58 copy procedure exit complement
incA ; 110 08 59 increment ax (address in daughter to copy to)
incB ; 110 09 60 increment bx (address in mother to copy from)
jmpo ; 110 14 61 bidirectional jump to template below (copy loop)
nop0 ; 100 00 62 copy loop complement
nop1 ; 100 01 63 copy loop complement
nop0 ; 100 00 64 copy loop complement
nop1 ; 100 01 65 copy loop complement
ifz ; 000 05 66 this is a dummy instruction to separate templates
nop1 ; 110 01 67 copy procedure exit template
nop0 ; 110 00 68 copy procedure exit template
nop1 ; 110 01 69 copy procedure exit template
nop1 ; 110 01 70 copy procedure exit template
popC ; 110 12 71 pop cx off stack (size)
popB ; 110 11 72 pop bx off stack (start address of mother)
popA ; 110 10 73 pop ax off stack (start address of daughter)
ret ; 110 17 74 return from copy procedure
nop1 ; 100 01 75 end template
nop1 ; 100 01 76 end template
nop1 ; 100 01 77 end template
nop0 ; 100 00 78 end template
ifz ; 000 05 79 dummy instruction to separate creature

```

Appendix B: A New Form of Informational Parasitism

Here is an example of a host and its parasite that uses host code to calculate its own size, adjust its starting address @ in register *ax*, allocate space for its daughter, in addition to using the host's copying loop. (See main text for discussion of this metabolic informational parasitism.)

```
PARASITE (Metabolic Informational Parasite)
format: 3 bits: 3 EX TC TP MF MT MB
genotype: 0020afe parent genotype: 0020afc
1st_daughter: flags: 0 inst: 159 mov_daught: 20 breed_true: 1
2nd_daughter: flags: 0 inst: 156 mov_daught: 20 breed_true: 1
Origin: InstExe: 11994,829667 clock: 1014497598 Sat Feb 23 15:53:18 2002
MaxPropPop: 0.0588 MaxPropInst: 0.0253 mpp_time: 12004,193177
```

```
nopl ; 000 01 0 | start template
nopl ; 000 01 1 |
call ; 000 16 2 |
call ; 000 16 3 |
adrb ; 000 1c 4 }
nop0 ; 000 00 5 } put ax=@+2, cx=2
nop0 ; 000 00 6 } by matching start template
jmpb ; 000 15 7 |
nopl ; 000 01 8 | Jump to host
nopl ; 000 01 9 |
nop0 ; 000 00 10 |
call ; 000 16 11 |
nopl ; 000 01 12 |
nopl ; 000 01 13 |
nop0 ; 000 00 14 |
ret ; 000 17 15 |
popB ; 000 11 16 |
nopl ; 000 01 17 |
nopl ; 000 01 18 |
nopl ; 000 01 19 |
```

```
Independently Replicating HOST
format: 3 bits: 3 EX TC TP MF MT MB
genotype: 0036alf parent genotype: 0036alf
1st_daughter: flags: 0 inst: 271 mov_daught: 36 breed_true: 1
2nd_daughter: flags: 0 inst: 267 mov_daught: 36 breed_true: 1
Origin: InstExe: 2895,924123 clock: 1014452739 Sat Feb 23 03:25:39 2002
MaxPropPop: 0.6702 MaxPropInst: 0.3492 mpp_time: 11855,322749
```

```
nopl ; 000 01 0 |
nopl ; 000 01 1 |
nop0 ; 000 00 2 |
call ; 000 16 3 |
call ; 000 16 4 |
adrb ; 000 1c 5 |
nop0 ; 000 00 6 | parasite sends control to
nop0 ; 000 00 7 | host code by matching this template
nopl ; 000 01 8 |
subAAC ; 000 07 9 subtract template size cx from ax=@+cx, put result in ax
movBA ; 000 19 10 put start address ax of (host or parasite) in register bx
shl ; 000 03 11 |
shl ; 000 03 12 |
decC ; 000 0a 13 |
not0 ; 000 02 14 | Size calculating code produces 36 if cx=3 at line 9
shl ; 000 03 15 | (but produces 20 if cx=2 at line 9
shl ; 000 03 16 | parasite's entry point).
decC ; 000 0a 17 | result is cx=36 for host
decC ; 000 0a 18 | or cx=20 for parasite
decC ; 000 0a 19 |
not0 ; 000 02 20 |
mal ; 000 1e 21 allocate space for daughter (size cx )
nopl ; 000 01 22 |
nop0 ; 000 00 23 |
movii ; 000 1a 24 |
decC ; 000 0a 25 |
ifz ; 000 05 26 |
jmpo ; 000 14 27 |
nop0 ; 000 00 28 |
incA ; 000 08 29 |
incB ; 000 09 30 |
jmpb ; 000 15 31 |
nop0 ; 000 00 32 |
nopl ; 000 01 33 |
divide ; 000 1f 34 |
ret ; 000 17 35 |
```

Appendix C: Smallest Known Independently Replicating Tierrans

Here is the smallest known independently self-replicating Tierran (0023c1n) using arithmetic operators to determine its own size:

```
format: 3 bits: 3 EX TC TP MF MT MB
genotype: 0023c1n parent genotype: human
1st_daughter: flags: 1 inst: 151 mov_daught: 23 breed_true: 1
2nd_daughter: flags: 0 inst: 150 mov_daught: 23 breed_true: 1
Origin: InstExe: 0,0 clock: 0 Thu Jan 1 01:00:00 1970
MaxPropPop: 0.8232 MaxPropInst: 0.4542 mpp_time: 0,349001
```

```
divide ; 000 1f 0
nop0 ; 000 00 1
adrb ; 000 1c 2
nop1 ; 000 01 3
incC ; 000 0b 4
subAAC ; 000 07 5
pushA ; 000 0c 6
incC ; 000 0b 7
shl ; 000 03 8
shl ; 000 03 9
shl ; 000 03 10
decC ; 000 0a 11
movBA ; 000 19 12
mal ; 000 1e 13
nop0 ; 000 00 14
decC ; 000 0a 15
movii ; 000 1a 16
ifz ; 000 05 17
ret ; 000 17 18
incA ; 000 08 19
incB ; 000 09 20
jmpb ; 000 15 21
nop1 ; 000 01 22
```

Here is the smallest known independently self-replicating Tierra (18c1n). This one uses the standard self-measuring mechanism to determine its own length:

```
format: 3 bits: 3 EX TC TP MF MT MB
genotype: 0018c1n parent genotype: human
1st_daughter: flags: 1 inst: 116 mov_daught: 18 breed_true: 1
2nd_daughter: flags: 0 inst: 114 mov_daught: 18 breed_true: 1
Origin: InstExe: 0,0 clock: 0 Thu Jan 1 01:00:00 1970
MaxPropPop: 0.8249 MaxPropInst: 0.4557 mpp_time: 0,338570
```

```
adrb ; 000 1c 0
nop1 ; 000 01 1
divide ; 000 1f 2
pushA ; 000 0c 3
movBA ; 000 19 4
adrf ; 000 1d 5
nop1 ; 000 01 6
subCAB ; 000 06 7
mal ; 000 1e 8
nop1 ; 000 01 9
decC ; 000 0a 10
movii ; 000 1a 11
ifz ; 000 05 12
ret ; 000 17 13
incA ; 000 08 14
incB ; 000 09 15
jmpb ; 000 15 16
nop0 ; 000 00 17
```

The Tierran 0018c1n replicates successfully when alone since it is preceded by nop0 (empty spaces) in the soup. It can also replicate successfully socially (when preceded by a copy of itself, or anything ending in nop0). Similarly, the size 0023c1n replicates successfully. Sizes 24 and 20 were the dominant sizes of descendents of 23c1n and 18c1n, respectively, in soups inoculated with a single individual.

These two Tierrans were designed by the author based on evolved forms.