DOI: http://dx.doi.org/10.7551/978-0-262-31709-2-ch032

Evolution of G-P mapping in a von Neumann Self-reproducer within *Tierra*

Declan Baugh, Barry McMullin

The Rince Institute, Dublin City University, Ireland
declan.baugh2@mail.dcu.ie, barry.mcmullin@dcu.ie

Abstract

John von Neumann first presented his theory of machine self-reproduction in the late 40's (von Neumann, 1948), in which he described a machine capable of performing the logical steps necessary to accommodate self-reproduction. The proposed architecture was comprised of two distinct components, a passive genotype, which acts exclusively as an information storage of a machine description, and an active phenotype which is responsible for all mechanical functionality of the machine including the ability to decode the genotype and construct the described machine to facilitate self-reproduction. This paper presents an exploratory model which implements the von Neumann architecture for self-reproduction within the pre-existing evolutionary platform of Tierra. Initially, the memory image of the automaton's genotype and phenotype are physically identical, and each symbol in memory may be interpreted as either passive numerical data (g-symbol), or a functional instruction (p-symbol), depending on how the symbol is interpreted. If redundancy is introduced to a mutable genotype-phenotype mapping, the mapping system becomes non-invertible, rendering it impossible to compute an automaton's exact genotypic memory image by analysis of the phenotype alone. However, this non-invertible mapping may allow for a more robust genotype, increasing its robustness to fatal mutations and therefore increasing its ability to preserve its phenotypic form under perturbations.

The von Neumann Architecture for Machine Self-reproduction

Von Neumann's architecture for machine self-reproduction, presented in his theory of self-reproducing automata (von Neumann, 1966; Baugh and McMullin, 2012), describes a machine, M, which is decomposed into two primary components, a functional component P, and an passive component G, such that $M = (P + G)$ (McMullin, 2012). G represents a one-dimensional string of symbols which has no active/functional capability, but can be interpreted as information, similar to the *tape* of a Turing machine. The information within G is used to describe an arbitrary machine X under some function, $\phi()$, such that $G = \phi(X)$.

P is further divided into four fundamental subcomponents, a *general constructive automaton* A, a *general copying automaton* B, a *control unit* C, and the *ancillary machinery*, D. G will be referred to as the *genotype* while P will be referred to as the *phenotype*[1].

The general constructive automaton A can read the symbols within G, and interpret them as an encoded description of an arbitrary machine X. A has the capability to apply an inverse function, $\phi^{-1}()$, or $\psi()$, to G, and construct the described machine X. We denote this by saying $\psi(G) = \psi(\phi(X)) = \phi^{-1}(\phi(X)) = X$. In other words, when supplied with a genotype, the general constructive automaton applies the decoding function $\psi()$, to G, in order to construct the arbitrary machine X.

The general copying automaton B, reads and duplicates the machine description $\phi(X)$. A control unit C is required to govern the automaton $(A + B)$, directing its operation, activating A and B in the correct order, and insuring that the offspring creature is "activated" once its construction is complete.

The forth component, the ancillary machinery D, refers to all conceivable functionality that the machine may possess which does not interfere or hinder the reproductive operation of $(A + B + C)$.

When a machine $(A + B + C + D)$ is supplied with a description $\phi(X)$, the control unit C first commands B to duplicate $\phi(X)$. Upon duplication, C instructs A to decode $\phi(X)$ under some inbuilt genotype-phenotype mapping function $\psi()$, and construct the described machine X. Finally C will attach the new instances of $\phi(X)$ and X, and sever them from the parent automaton $(A + B + C + D)$, after which there exists the new entity, $X + \phi(X)$. Now consider the case where $X = (A + B + C + D)$. This system, $(A + B + C + D) + \phi(A + B + C + D)$ will proceed to construct an offspring automaton and attach it to the description of itself, $(A + B + C + D) + \phi(A + B + C + D)$. The parent and offspring are identical, therefore achieving self-reproduction. This machine architecture is demonstrated in Figure 1.

[1] Although von Neumann never used these terms, we now associate the components in question with the genotype and phenotype in organic biology.

Next we consider the case where a random phenotypic perturbation occurs during the construction of P, affecting D, so that a machine $M = (P + G)$ produces $M' = (P' + G)$, where $P' = (A + B + C + D')$. This machine $(A + B + C + D') + \phi(A + B + C + D)$ will proceed to decode and copy the unaltered G under a decoding function $\psi()$, to recreate the original machine $M = (A + B + C + D) + \phi(A + B + C + D)$, and the phenotypic perturbation is not inherited. For the case where the perturbation affects the description of D in G, creating a machine $M' = (A + B + C + D) + \phi(A + B + C + D')$, M' will proceed to decode and copy G' under a decoding function $\psi()$ to create a new machine, $M'' = (A + B + C + D') + \phi(A + B + C + D')$, where $M'' = (P' + G')$[2]

Should a random perturbation occur when copying the description of A to an offspring, which results in $\phi(A' + B + C + D)$, then the machine $(A+B+C+D)+\phi(A'+B+C+D)$ will produce $(A'+B+C+D)+\phi(A'+B+C+D)$. It is possible that this machine will now have an altered general constructive automaton. Von Neumann/Burks stated "If the change is in A, B or C, the next generation will be sterile." (von Neumann, 1966, p. 86), however it is conceivable that a perturbation within the description of A may only affect the decoding function $\psi()$ without completely breaking its reproductive functionality.

This machine $(P' + G')$ may not be sterile, but conduct the altered decoding function $\psi^*()$ where $\psi^*(G') = \psi^*(\phi(P')) = (P')^*$, to construct the machine $((P')^* + G')$. If there are changes in the mapping which allow $(P')^* = P'$, then the machine $(P' + G')$ will self-reproduce successfully while conducting a different genotype-phenotype mapping (McMullin, 2000). We aim to scrutinise this intricate detail, and investigate if any additional mutational pathways may emerge as a result of a mutable genotype-phenotype mapping.

Implementation within the Tierra Platform

Tierra is an artificial life platform where populations of assembler language self-reproducing automatons (creatures) compete with one another within a one-dimensional circular core memory for both CPU time and memory space (Ray, 1991). Each Tierran automaton consists of a CPU with up to six registers, stack memory and an instruction pointer.

Typically, self-reproduction within Tierra is accomplished via *self-copying*, where a creature must inspect its entire memory image in order to construct an identical offspring. This mechanism is loosely analogous to the reproduction process which occurs in the RNA world hypothesis which posits that at the earlier stages of evolution, RNA acted as both template and template-directed polymerase, and there existed no distinction between genotype and phenotype.

In order to implement the von Neumann architecture within the platform of Tierra, the seed automaton must enforce a division of labour between the storage of genetic information and the catalytic functionality, hence recognising the roles of genotype and phenotype.

The phenotype will naturally consist of the three subcomponents, a general constructive automaton A, a general copying automaton B, and a control unit C. The control unit segment of the automaton will calculate the offspring size and allocate memory space to construct the offspring. The general constructor segment will incorporate a mutable genotype-phenotype mapping to allow for inheritable variation which may result in new evolutionary trajectories with creatures conducting an altered genotype-phenotype mapping. The general constructor will incrementally read the symbols within the genotype and under some genotype-phenotype mapping, will determine which p-symbols are to be written to the offspring phenotype. Upon construction of the offspring phenotype the copier is activated and the g-symbols within the parent's genotype are incrementally read and written to the offspring genotype. The control unit then activates the offspring automaton, and the reproductive cycle repeats.

mRNA-Amino Acid Inspired Genotype-phenotype Mapping

In order to encode the phenotype of an automaton, an arbitrary genotype-phenotype mapping must be implemented. The evolutionary trajectory of such an automaton will be in part, determined by the nature of this arbitrarily elected mapping. We can only claim that any phenomenon observed will be specific and characteristic to the specific mapping system which is implemented. For the purpose of this project, a bijective, mono-alphabetic substitution cipher was chosen. This method was loosely based on the genetic code, in which an mRNA, consisting of a one-dimensional string of symbols (nucleotides), is transcribed into a specific string of symbols (amino acids). If a single letter in an mRNA codon gets perturbed, then the affected codon may result in the construction of a different amino acid.

If we implement a similar mapping system which allows perturbations to the genotype which may alter the description of the general constructor, specifically, altering the genotype-phenotype mapping function $\psi()$, then we would effectively be implementing a von Neumann reproducer which may give rise to new evolutionary trajectories operating an altered genotype-phenotype mapping.

This type of a mutable genotype-phenotype mapping was facilitated via the inclusion of a lookup table within the general constructor. The lookup table consists of a one-dimensional string representing the full list of p-symbols available within the phenotype space.

[2] It is worth noting that when a perturbation occurs within P, the perturbation is not inherited in further generations, however when the perturbation occurs within G, there is a generation delay between when the perturbation occurs in the genotype and when it is expressed in the phenotype.

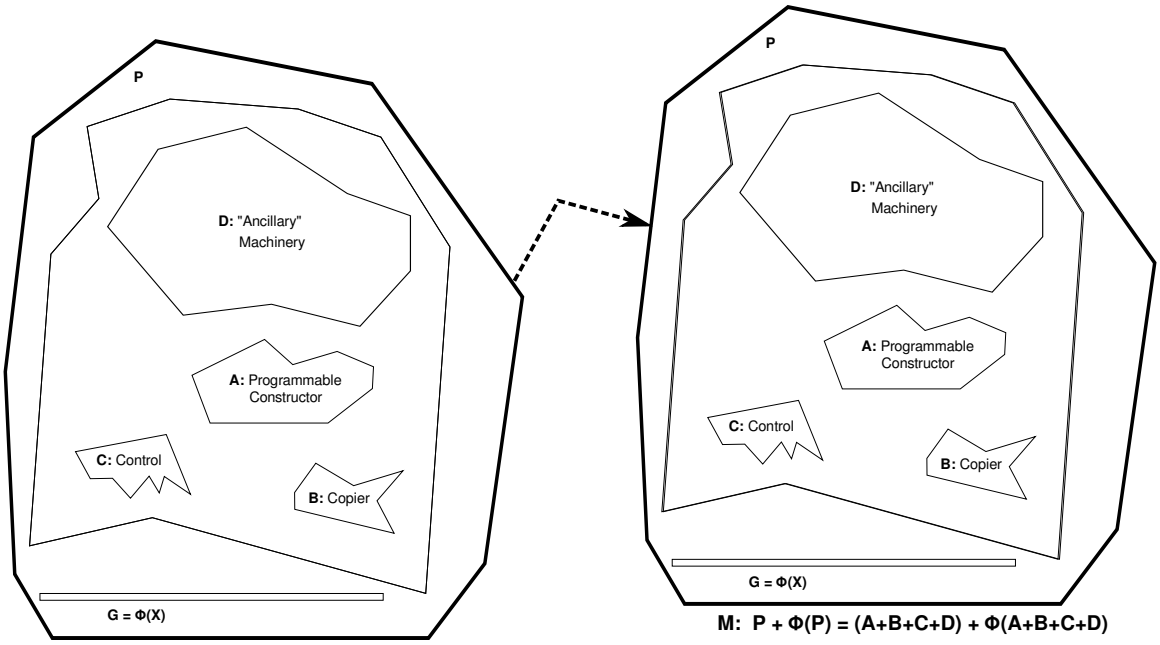

Figure 1: A schematic of the von Neumann style architecture of machine self-reproduction. Excerpted from McMullin (2012).

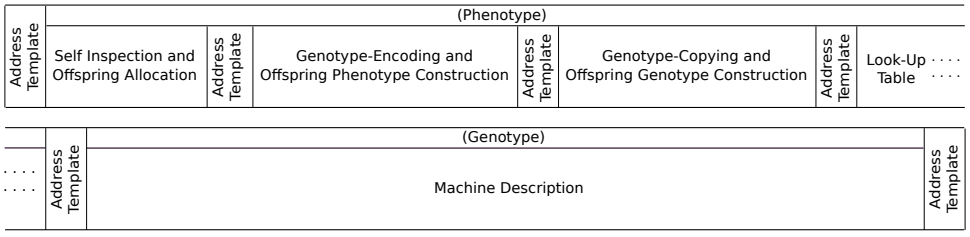

Figure 2: A schematic of a von Neumann style ancestor in Tierra. The genotype-encoding and offspring phenotype construction, and the lookup table comprises the general constructive automaton. The genotype-copying and offspring genotype construction segment comprises the general copying automaton, and the self-inspection and offspring allocation segment comprises the control unit.

During construction of an offspring phenotype, the parent's genotype is incrementally examined by the general constructor. The g-symbol at a memory location is read and interpreted as its underlying binary numerical value. The p-symbol situated at the corresponding relative address in the lookup table is read, and is written to the offspring's phenotype where it will function as an active instruction. The general constructor sequentially executes this process for every location in the genotype in order to decode each g-symbol and construct the offspring phenotype.

In an attempt to replicate conditions early in the phase change from simple RNA replicators to a system of mRNA and amino acids, we implemented an identity mapping from genotype to phenotype, where the content of the one-dimensional memory image of the genotype is physically identical to that of the phenotype. This situation is analogous to that of the RNA world hypothesis, where in order to replicate, an RNA molecule can act as both a functional catalyst, or a string of symbols to be interpreted, depending on whether it is acting as the catalyst or template.

In cryptography we would refer to this identity mapping as being an encryption in which the plaintext is identical to the ciphertext. Plaintext is information a sender wishes to transmit to a receiver and ciphertext is the result of encryption performed on the plaintext using an algorithm, called a cipher. In this case, the phenotype can be thought of as an un-encrypted message which is to be transmitted to the offspring's memory image and the genotype is the ciphertext which was a result of encrypting the phenotype, according to the encoding function $\phi()$. The initial decoding function $\psi()$ is determined by the permutation of symbols within the lookup table. It is worth noting however, that regardless of the initial permutation chosen for the lookup table, the initial description of the lookup table will always remain the same. If we let S represent the non-permutated list of symbols which exist in the Tierra universe[3], then the permutation of the lookup table depicts how each individual element within this non-permutated list of symbols is decoded under $\psi()$[4]. The lookup table can now be described as $\psi(S)$. When we encode a phenotype to acquire the genotype, the encoded lookup table will now be represented by $\phi(\psi(S)) = \psi^{-1}(\psi(S)) = S$, therefore, regardless of the initial permutation of the lookup table, the initial description of the lookup table will always take the form of S.

Our ancestor requires a minimum of 28 phenotypic instructions in order to self-reproduce. The mRNA-amino acid transcription table consists of a genotype space of 64 different codons, but a significantly smaller phenotype space of

UUU	(Phe/F)	UCU	(Ser/S)	UAU	(Tyr/Y)	UGU	(Cyc/C)
UUC		UCC		UAC		UGC	
UUA	(Leu/L)	UCA		UAA	Stop(Ochre)	UGA	Stop (Opal)
UUG		UCG		UAG	Stop(Amber)	UGG	(Trp/W)
CUU		CCU	(Pro/P)	CAU	(His/H)	CGU	(Arg/R)
CUC		CCC		CAC		CGC	
CUA		CCA		CAA	(Gln/Q)	CGA	
CUG		CCG		CAG		CGG	
AUU	(Ile/I)	ACU	(Thr/T)	AAU	(Asn/N)	AGU	(Ser/S)
AUC		ACC		AAC		AGC	
AUA		ACA		AAA	(Lys/K)	AGA	(Arg/R)
AUG	(Met/M)	ACG		AAG		AGG	
GUU	(Val/V)	GCU	(Ala/A)	GAU	(Asp/D)	GGU	(Gly/G)
GUC		GCC		GAC		GGC	
GUA		GCA		GAA	(Glu/E)	GGA	
GUG		GCG		GAG		GGG	

00000	(nop0)	10000	(decC)	100000	(popC)	110000	(movAb)
00001	(nop1)	10001	(nop10)	100001	(nop19	110001	(nop29)
00010	(nop2)	10010	(incD)	100010	(popD)	110010	(movda)
00011	(nop3)	10011	(nop11)	100011	(nop20)	110011	(nop30)
00100	(ifnz)	10100	(pushA)	100100	(popE)	110100	(movAb)
00101	(nop4)	10101	(nop12)	100101	(nop21)	110101	(nop31)
00110	(addAAE)	10110	(nop13)	100110	(nop22)	110110	(ret)
00111	(nop5)	10111	(nop14)	100111	(nop23)	110111	(nop32)
01000	(subCAB)	11000	(pushC)	101000	(jmpb)	111000	(nop33)
01001	(nop6)	11001	(nop15)	101001	(nop24)	111001	(shlA)
01010	(subAAC)	11010	(pushD)	101010	(adrf)	111010	(nop34)
01011	(nop7)	11011	(nop16)	101011	(nop25)	111011	(nop35)
01100	(incA)	11100	(popA)	101100	(nop26)	111100	(mal)
01101	(nop8)	11101	(nop17)	101101	(nop27)	111101	(nop36)
01110	(incB)	11110	(popB)	101110	(call)	111110	(nop37)
01111	(nop9)	11111	(nop18)	101111	(nop28)	111111	(divide)

Figure 3: The upper figure presents the mapping from mRNA to amino acid. The lower figure presents the initial mapping from g-symbols to p-symbols, implemented with our von Neumann style seed automaton. Red symbols highlight those which are non-employed and grey highlights symbols which initially map onto employed p-symbols.

22 amino acids, plus the start and stop codons. In an attempt to mirror the redundancy of the genetic code, a genotype and phenotype space of 64 was implemented. This corresponds to 64 separate 6-bit binary digits in the genotype space, and 64 phenotypic instructions, 28 of which have an active function and are used within the phenotype and contribute towards self-reproduction, and 36 of which have no active function at all. (See Figure 3.)

Experimental Procedure

The Tierra soup was inoculated with the described von Neumann style ancestor, (Figure 2).

Point perturbations which affect random memory locations throughout the soup (cosmic rays) and perturbations which occur exclusively to symbols that are being written to memory locations in the soup (copy perturbations) were enabled and the system was run for 100 billion CPU cycles, which is approximately 250 thousand generations[5]. Every strain of creature that emerged throughout the run was captured and the number of employed and non-employed p-symbols within the lookup table for each creature was

[3] In this case, the non-permutated list of symbols is represented by a list of consecutive binary numbers from 000000 to 111111.

[4] The first location in the lookup table represents which p-symbol is mapped onto by 000000. The second position in the lookup table represents which p-symbol is mapped onto by 000001 etc.

[5] A generation in Tierra is a calculated time interval, which is determined by the estimated average amount of CPU cycles required for each creature present in the soup to reproduce once and die.

counted. Employed p-symbols refer to those which have a functional role in the process of reproduction, while non-employed p-symbols are included to introduce redundancy and do not actively contribute towards the reproduction process. If a specific p-symbol exists in the lookup table, then there must exist a specific g-symbol which maps onto it. If a p-symbol is absent from the lookup table then it is lost from the genotype-phenotype mapping. With our current mapping system[6], it is impossible for a p-symbol which is absent in a parents lookup table to be included in its offspring's phenotype (with the exception of random phenotypic perturbations introducing random p-symbols to an offspring).

The population of employed vs. non-employed p-symbols in the lookup table of each creature was then plotted against the time of emergence of that individual, and this process was repeated 4 times.

Results

Standard evolutionary Behaviour

Initial simulations showed evolutionary behaviour similar to that documented in Ray's initial experiments (Ray, 1991). Informational parasitism[7] quickly emerged due to the description of the lookup table being omitted from the genotype. The resulting creature will redirect its CPU to a neighbouring host to facilitate the construction of its phenotype and therefore expend less CPU time per reproduction cycle due to its reduced length. Another evolutionary phenomena typical of Rays experiments is the reduction of creature size by reducing template addresses where possible. Address locations in Tierra are not facilitated via a global address location, but by matching complementary patterns of `nop1`'s and `nop0`'s. While the programmer creating the creature may use an initial template size of four `nop` instructions, evolution will typically reduce the template size where ever possible, creating shorter and more efficient offspring. Other evolutionary behaviours observed when implementing von Neumann style reproduction are the emergence of pathological constructors (Baugh and McMullin, 2012) and the degeneration to self-copiers within in the platform of Avida (Hasegawa and McMullin, 2012).

Evolution of the Genotype-phenotype mapping

The aforementioned evolutionary behaviours have already been studied and documented, and therefore is not of primary concern, so for the remaining experiments the input parameters were edited so that only offspring of the same length as the initial ancestor are allowed to propagate throughout the memory. This will prevent the distraction of known phenomenon occurring and allow us to focus on the specific evolutionary lineages which arise as a direct result of a change in the genotype-phenotype mapping. A change in the genotype-phenotype mapping will be most easily recognised by a change in the lookup table.

Initially, non-fatal inheritable silent perturbations[8] of the genotype will occur in the description the lookup table. This will alter the genotype-phenotype mapping and allow previously silent g-symbols[9] to be mapped onto employed p-symbols. This allows single employed p-symbols to be mapped onto by multiple g-symbols.

The initial ancestor has 36 silent g-symbols, which are mapped onto 36 different non-employed p-symbols. As neither the silent g-symbols nor the non-employed p-symbols functionally contribute to the reproduction of offspring, the silent mutations that affect which p-symbol the silent g-symbols are mapped onto are random and arbitrary. However it was found that there was a strong bias towards the mapping of silent g-symbols onto employed p-symbols. During an evolutionary run, we see a sharp decrease in the number of non-employed p-symbols within the lookup tables of newly emerging creatures. Eventually, all 36 non-employed p-symbols are eliminated from the descendants of the initial ancestor, and the 64 positions in their lookup tables will consist almost entirely of employed p-symbols. This result can be seen in Figure 4.

Discussions

The evolution of the genotype-phenotype mapping will initially be driven predominantly by the underlying physical dynamics of the coding system. The nature of the substitution cypher mapping mechanism employed means that certain perturbations of the lookup table are not directly reversible. This results in a biased drift in the genotype-phenotype mapping, eventually eliminating all non-employed p-symbols from the phenotype by ensuring that they are not mapped onto by any elements of the genotype space.

Figure 5 demonstrates a small section of the lookup table and its description. By studying a creature's lookup table and the lookup table's description, one can deduce the genotype-phenotype mapping that is implemented by that creature. For this small section of the mapping between the g-symbols and p-symbols, we see a set of four g-symbols which are interpreted as `0`, `1`, `2` and `3`, which are mapped onto four p-symbols which are interpreted as `nop0`, `nop1`, `nop2` and `nop3` respectively. The red symbols within the lookup table represent non-employed p-symbols, while the grey symbols within the lookup table's description represent the silent g-symbols which initially map onto a non-employed p-symbol. Figure 5(a) demonstrates the initial 1st

[6] A mono-alphabetic substitution cipher.

[7] Informational parasitism refers to a form of parasitism which accesses and reads a host's memory contents, but does not directly interfere with its functionality.

[8] A silent perturbation is one which alters the genotypic sequence, but does not affect the structure of the phenotype.

[9] By silent g-symbols, we refer to g-symbols which were initially mapped onto non-employed p-symbols.

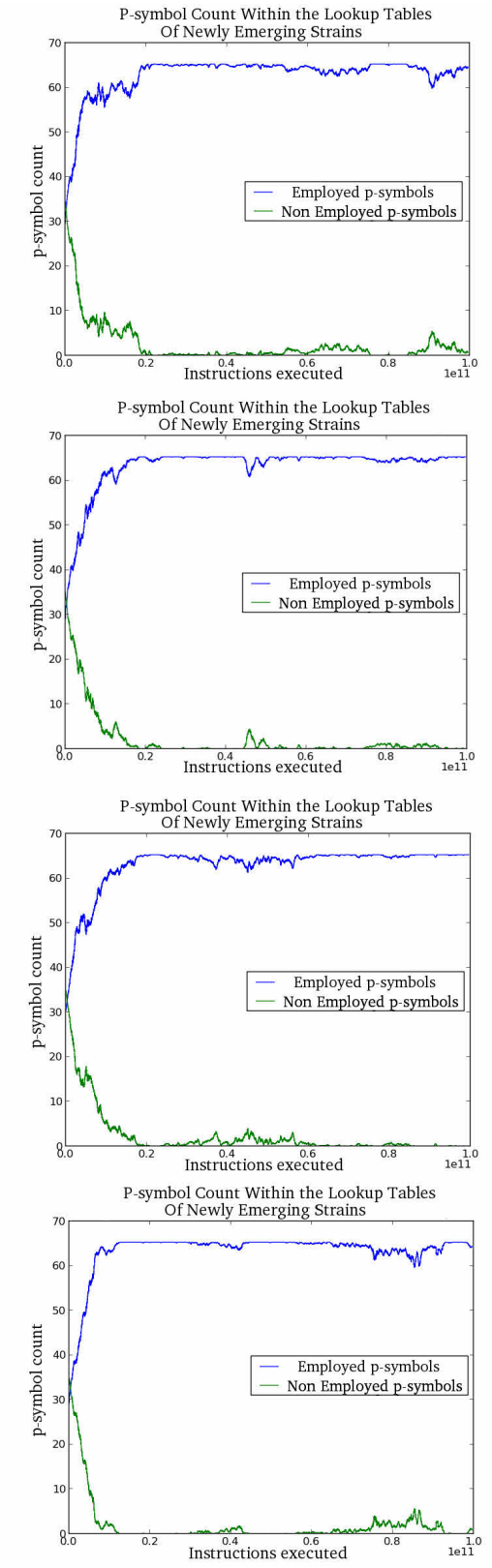

Figure 4: Four evolutionary simulations displaying the number of employed vs. non-employed p-symbols present in the lookup table of strains of newly emerging lineages.

generation ancestor's lookup table and description. We can see here that the mapping is initially injective and surjective (bijective), as each element of the genotype space is mapped onto a different element of the phenotype space. This mapping is also invertible, as it is possible to determine the exact genotypic sequence by analysis of the phenotype alone. This can be denoted by saying that $P \triangleq \phi(G)$ and $G \triangleq \psi(P)$ where $\psi() = \phi^{-1}()$.

Figure 5(b) represents an offspring which experienced a perturbation to the third position in the lookup table's description. For von Neumann reproduction, there is a generation delay between when a perturbation occurs in a genotype and when the perturbation is expressed in the phenotype. When this 2nd generation creature attempts to reproduce, it must first copy its exact genotype to the 3rd generation offspring Figure 5(c). The 2nd generation creature must then decode its own genotype, and construct the 3rd generation creature's phenotype. However, under construction of the phenotype, when decoding the 2nd position[10] in the lookup table description, the employed p-symbol, `nop0`, is written to the third position in the lookup table, and the previous non-employed p-symbol, `nop2`, is lost from the genotype-phenotype mapping.

Even if the perturbed position in the lookup table description gets perturbed back to the previous state, Figure 5(d), the non-employed p-symbol does not return to the phenotype. This is because the genotype-phenotype mapping has been changed, and now the silent g-symbol, which initially was mapped onto a non-employed p-symbol, `nop2`, is now mapped onto an employed p-symbol, `nop0`.

We also see that the mapping is now non-injective and non-surjective, as an element of the phenotype space, `nop0`, is mapped onto more than one element of the genotype space, `0` and `2`. Furthermore, an element of the phenotype space, `nop2`, is not mapped onto by any element of the genotype space. This renders the mapping invertible, as it is now impossible to determine the exact genotypic sequence via inspection of the phenotype alone as $\psi() \neq \phi^{-1}()$ as now $G = \psi^*(P)$.

The only method in which a lost non-employed p-symbol can return to the lookup table is via a genotypic perturbation, which returns the lookup table description to its previous state, followed by a phenotypic perturbation, which directly introduces the lost non-employed p-symbol to the lookup table. Due to this level of ease at which a non-employed p-symbol can be lost, and the level of difficulty required to re-introduce the non-employed p-symbol to the mapping, there is a strong immediate bias present, which quickly eliminates all non-employed p-symbols from the lookup table.

This feature of the implemented mapping system demonstrates how *phenotypic* perturbations are inheritable under the circumstance that the perturbation affects the function

[10] Using zero-based indexing.

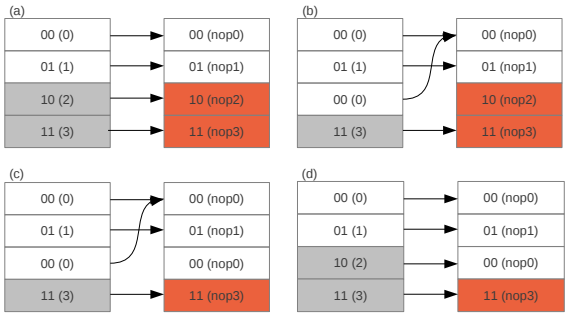

Figure 5: Perturbations to the lookup table description

$\psi()$. If we have a machine $X = (A' + B + C + D) + \phi(A+B+C+D)$, where A' represents a general constructor with a changed genotype-phenotype mapping $\psi^*()$, then this perturbation will be inherited to the offspring phenotype only if $\psi^*(G') = \psi(G')$, where $G' = \phi(A' + B + C + D)$. In other words, this demonstrates an instance of Lamarkian inheritance, where a perturbation of the phenotype is passed down to future generations without any change to the genotype. However, in order for this to occur, the perturbation must affect the component of the phenotype which decodes the genotype, such that the genotype will now be decoded to give rise to the new perturbed phenotype.

Mutational robustness and Darwinian selection. Darwinian selection may then sharpen the genotype-phenotype mapping, and create a more mutationally robust genotype. The allocation in which silent g-symbols are mapped upon employed p-symbols may be subject to Darwinian selection. Following a perturbation to a g-symbol, a phenotype may still preserve form if both g-symbols transcribes to the same p-symbol.

If there was no redundancy in the genotype space, then the mapping cannot incorporate inherent mechanism to ensure stability to perturbations and help the phenotype preserve its form under inheritable variation. Every p-symbol will have the same robustness to mutation, no matter how frequently its description occurs in the genotype, or how imperative it is to the correct operation of the phenotype.

For our experiments, we only have 28 employed p-symbols, but 64 g-symbols. Darwinian selection may select how the silent g-symbols are mapped upon the employed p-symbols. A p-symbol which is very common within the phenotype, has a high probability of having its description perturbed within the genotype. If a large percentage of the silent g-symbols are mapped upon the most frequent, employed p-symbols, then the phenotype will have an increased probability of holding form following an inheritable perturbation to the genotype. To test this hypothesis, a creature was engineered with a non-surjective genotype-phenotype mapping. All silent g-symbols were mapped onto the employed p-symbol, `nop0`. `nop0` is very frequent throughout the phenotype, as it is used for template addressing. The mutational robustness of `nop0`'s description has now greatly increased, as there are 36 possible perturbations which will still allow the phenotype to preserve form. The Tierra soup was inoculated with two von Neumann self-reproducers, the original ancestor with a surjective genotype-phenotype mapping, and the engineered ancestor with the non-surjective genotype-phenotype mapping. The two creatures used different address templates, so that the descendants of each ancestor could be distinguished from each other. This simulation was run for 100 billion instructions and the experiment was repeated 100 times. It was found that in 76 instances the initial ancestor was driven to extinction, while in 24 instances the engineered ancestor with the non-surjective genotype-phenotype mapping was driven to extinction. These preliminary tests show that there may be a selective advantage for distributing the silent g-symbols amongst the most frequently occurring employed p-symbols, and therefore room for Darwinian selection to guide the evolution of the genotype-phenotype mapping. However, this work is pending and requires further experimentation.

The Tierra source code for these experiments along with the analysing software can be found at: `http://alife.rince.ie/evosym/alife_2013_dbbm.zip`.

Acknowledgements. This work has been supported by the European Complexity Network (Complexity-NET) through the Irish Research Council for Science and Technology (IRCSET) under the collaborative project EvoSym. Computing facilities were facilitated by The Irish Centre for High-End Computing (ICHEC).

References

Baugh, D. and McMullin, B. (2012). The emergence of pathological constructors when implementing the von neumann architecture for self-reproduction in tierra. In *From Animals to Animats 12*. Springer.

Hasegawa, T. and McMullin, B. (2012). Degeneration of a von neumann self-reproducer into a self-copier within the avida world. In *From Animals to Animats 12*, pages 230–239. Springer.

McMullin, B. (2000). John von neumann and the evolutionary growth of complexity: Looking backward, looking forward. *Artificial Life*, 6(4):347–361.

McMullin, B. (2012). Architectures for self-reproduction: Abstractions, realisations and a research program. In *Artificial Life 13*, pages 83–90.

Ray, T. (1991). An approach to the synthesis of life. *Artificial life II*, 10:371408.

von Neumann, J. (1948). The general and logical theory of automata. In *Cerebral Mechanisms in Behaviour*, pages 1–32.

von Neumann, J. (1966). *Theory of self-reproducing automata*. Edited and completed by A.W. Burks.