

Contents

1	Einleitung	1
2	Prämissen: Geschichte und Definitionen	2
2.1	T.S. Ray	2
2.2	Defintion: Leben	2
2.3	Definition: Evolution	2
2.4	Motivation	2
3	Tierra	3
3.1	Grundlegendes	3
3.2	Das Betriebssystem	3
3.2.1	Zuweisung Rechenzeit: der Slicer	3
3.2.2	Künstlicher Tod: der Reaper	3
3.2.3	Speicherzuweisung	4
3.2.4	Genetische Operatoren	4
3.2.5	Mutation	4
3.2.6	Gen-Splicing	4
3.2.7	”Absichtliche” Fehler	5
3.2.8	Störungen	5
3.3	Arbeitspeicher: die Suppe (Soup)	5
3.4	CPU	5
3.5	Anweisungen	6
3.6	Network Tierra	7
4	Organismen in Tierra	7
4.1	Der Vorfahre (Ancestor)	7
4.2	Parasiten	7
4.3	Hyperparasiten	9
4.4	Soziale Hyperparasiten	9
4.5	Hyper-Hyper-Parasiten	9
5	Multizelluläres, künstliches Leben in Tierra	9
5.1	Idee	9
5.2	1. Implementation	10
5.2.1	Ergebnisse	10
5.3	2. Implementation	11
5.3.1	Ergebnisse	11
6	Fazit und Ausblick	12
7	Anhang	12
7.1	Befehlssatz von Tierra	12
7.2	Quellen	13

Tierra und multizelluläres künstliches Leben

Hannes Planatscher

December 20, 2001

Abstract

T.S. Ray, ein Evolutionsbiologe, entwickelte Anfang der Neunziger einen revolutionären Forschungsansatz im Gebiet der Evolutionsbiologie und des künstlichen Lebens. Er schuf ein digitales Biotop, im Speicher eines virtuellen Rechners, bevölkert von Lebewesen aus Befehlen in Maschinencode. In diesem Biotop entwickelten sich zuerst einfache Programme, zu komplexen Populationen aus Parasiten, Symbionten und vielen anderen interessanten Genossen, im ständigen Kampf um Speicherplatz und Rechenzeit. Ray sieht die Evolution 'in silicio' und die Evolution 'in vivo' als Instanzen desselben Phänomens. Zuerst kritisch beäugt gelten T.S. Rays Arbeiten heute als einige der wichtigsten Beweise für die darwinistische Evolutionstheorie.

1 Einleitung

Die Evolutionsbiologie versucht das Phänomen Evolution zu erforschen und zu verstehen, und zwar an einem einzigen Beispiel: dem Leben auf der Erde. Um die Dynamik der Evolution zu verstehen muss man, laut Ray, dieses Konzept erweitern. Nun ist es unmöglich Leben auf anderen Planeten zu beobachten. Somit kam er auf die Idee eine Instanz von Leben in einer künstlichen Umgebung zu erschaffen, und an diesem Beispiel die Evolution zu erforschen. Wichtig erscheint hier Rays Definition von Leben: "Ich würde etwas als lebend bezeichnen, wenn es sich selbst reproduzieren kann, und zu einer Evolution mit offenem Ende fähig ist." .

Ray entwickelte tatsächlich eine solche künstliche Umgebung, genannt "Tierra". Tierra ist ein virtueller Computer der seine eigene Maschensprache besitzt, und einen Hauptspeicher. Dieser Computer hat den Vorteil, dass er sehr unempfindlich gegenüber fehlerhaften Anweisungen ist, und nicht zum Absturz gebracht werden kann. Nun konnten im Speicher dieser Maschinen selbstreplizierende Organismen ausgesetzt werden, die durch bestimmte Mechanismen evolvierten und sich weiterentwickelten. Das Ergebnis waren erstaunliche Erkenntnisse zur klassischen Evolutionstheorie.

In dieser Arbeit werde ich zunächst auf die Motivation und Denkweise Thomas Ray's eingehen, dann die Funktionsweise Tierras beschreiben, und welche Erkenntnisse es uns liefert. Im 2.ten Teil dieser Ausarbeitung werde ich auf die Forschung über multizelluläres künstliches Leben mittels Tierra eingehen.

Ich möchte die Einleitung mit der Bemerkung abschliessen, dass in diesem Dokument ein Organismus im Sinn von Tierra oft auch als Prozess bezeichnet wird.

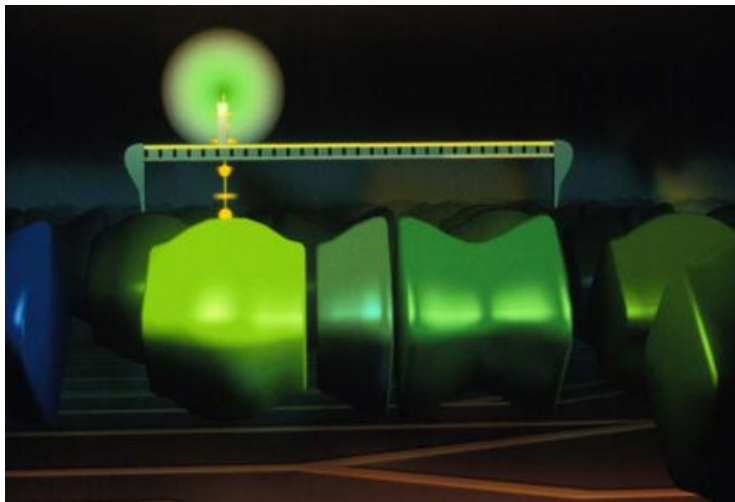


Figure 1: Darstellung eines tierranischen Organismus

2 Prämissen: Geschichte und Definitionen

2.1 T.S. Ray

Ray studierte zunächst Chemie und Biologie an der Florida State University. Er spielte mit dem Gedanken sich auch in der Physik und der Mathematik weiterzubilden, interessierte sich dann aber doch mehr für Ökologie, ganz im Sinne der sechziger Jahre, wie er zugibt. Ray spezialisierte sich auf Pflanzenökologie, und fertigte seine Doktorarbeit in Biologie an der Harvard University an. Ab 1974 ging er seinen evolutionsbiologischen Forschungen in den Regenwäldern von Costa Rica nach. Ab 1981 arbeitete an der University of Delaware. 1989 erhielt er eine Assistenz-Professur in Delaware. 1989 begannen auch seine Forschung, und sein spezielles Interesse für künstliches Leben. 1990 entwickelte er sein Programm Tierra. Im Zuge seiner Forschungen wurde er im August 1993 zum “Evolutionary System Department” am ATR (Advanced Telecommunications Research Institute International) in Japan eingeladen. 1998 erhielt er die Professur für Zoologie an der University of Oklahoma.

2.2 Definition: Leben

Leben wird meistens durch eine (mehr oder weniger lange) Reihe von ganz speziellen Eigenschaften definiert. T.S. Ray beschränkt sich in seiner Definition auf zwei Punkte, die seinen Hintergrund als Evolutionsbiologie widerspiegeln:

- Selbstreplikation
- Evolution mit offenem Ende

Diese Definition von Leben ermöglicht es, zwar unkörperliche aber echte, Instanzen von Leben in künstlichen Systemen zu finden. Daraus resultiert eine interessante, und mächtige Möglichkeit Leben und Evolution zu erforschen.

2.3 Definition: Evolution

Natürliche Evolution passiert in einem “physikalischen Medium”, und wird durch dieses konditioniert. Durch grosse Populationen und Selektion über viele Generationen, sucht Evolution die Möglichkeiten die der “Physik und Chemie” des Mediums inneligen. Das heisst, dass die Evolution im biologischen Medium vollkommen andere Ergebnisse und Wege finden wird, als Evolution im künstlichen, logischen Medium.

Vergleicht man Tierra mit genetischen Algorithmen liegt der grosse Unterschied darin, dass wir keine externe Fitnessfunktion festlegen müssen. Der Erfolg eines Organismus wird allein durch seine Fähigkeit bestimmt sich seiner Umwelt anzupassen, und seinen Fortbestand zu sichern.

2.4 Motivation

Warum ist es also wichtig und interessant Evolution in einem künstlichen Medium zu untersuchen?

- Beobachten neuer und unabhängiger Instanzen von Evolution
- Erweitern unseres Begriffs von Leben
- Erforschen einer natürlichen Evolution in einem digitalen Medium,

das sich radikal von dem bekannten physikalischen Medium unterscheidet.

3 Tierra

3.1 Grundlegendes

Tierra ist ein virtueller Rechner in dem natürliche Evolution passieren kann. Der Raum in dem Evolution passiert ist der Hauptspeicher(RAM). Die Lebensformen sind Computerprogramme aus Maschinencode. Bestimmte Eigenheiten des Betriebssystems, dem "Darwinian Operating-System" ermöglichen Evolution.

3.2 Das Betriebssystem

Das "Darwinian Operating-System" ist die Grundlage für die Evolution in Tierra. Es sorgt für die Zuweisung von Ressourcen an die Organismen, entfernen alter Organismen (künstlicher Tod), Speicherverwaltung, die genetische Operationen, und die biologische Unschärfe die für eine Evolution unbedingt nötig ist.

3.2.1 Zuweisung Rechenzeit: der Slicer

Der "Slicer" ist der Teil des Betriebssystems der den einzelnen Organismen (Prozessen) Rechenzeit der CPU zuweist. Er ist als Warteschlange aufgebaut, in der ein Organismus nach dem anderen, also seriell, bedient wird. Die Rechenzeit die einem Organismus zugewiesen wird bestimmt die Anzahl der Instruktionen die ausgeführt werden können. Die Länge dieses Zeitfensters kann konstant, zufällig oder in Abhängigkeit von der Grösse des Organismus gewählt werden. Wird ein neuer Organismus geboren wird er hinter dem Mutterorganismus in die Warteschlange eingefügt.

Von der Grösse des Zeitfensters hängt es ab ob später in der Evolution grosse oder kleine Organismen bevorzugt werden. Ist das Zeitfenster gross, werden grössere stabilere Organismen entstehen, da genügend Rechenzeit verfügbar ist, um diese vollständig zu kopieren. Sonst entstehen kleinere Organismen, die ihre Prozedur zur Reproduktion auf Schnelligkeit optimiert haben.

3.2.2 Künstlicher Tod: der Reaper

Tod ist in der Welt "Tierra" definiert als das Freimachen von Speicher (im Sinne vom Entfernen des Schreibschutzes vgl. 3.2.3), und das Entfernen eines Organismus aus der "Slicer"-Schlange.

Der "Reaper" sorgt dafür, dass alte Organismen entfernt werden und somit Speicherplatz freigeben. Der Reaper ist eine Schlange von Prozessen, mit fester Länge. Wird ein Organismus neu geboren wird er hinten an die Schlange angefügt. Der Reaper tritt in Aktion wenn:

- ein Organismus das Ende der Schlange erreicht hat
- ein Organismus zufällig aus x Organismen die sich Ende der Schlange befinden ausgewählt wird

Generiert ein Organismus einen Fehler wird er eine Position nach vorne geschoben. Pflanzte sich ein Organismus erfolgreich fort darf er in der "Warteschlange des Todes" eine Position zurück gehen.

3.2.3 Speicherzuweisung

Prozesse werden mit einem Speicherblock geboren. Speicher kann schreib-, lese- und/oder ausführungsgeschützt sein. Normalerweise ist Speicher schreibgeschützt damit Organismen nicht andere Organismen überschreiben können. Prozesse können nur einen weiteren Speicherblock für sich beanspruchen. Fordert der Prozess einen weiteren Block, wird der bereits zusätzlich zugewiesene, noch einmal zugewiesen.

Der Speicherverwalter kümmert sich um die Zuweisung der Speicherblöcke. Es gibt verschiedene Möglichkeiten um einen Speicherblock zu selektieren und ihm einen Prozess zuzuweisen. Der Speicherverwalter sucht die erstbeste freie Stelle im Speicher. Er kann auch die am besten passende Lücke im Speicher, oder eine freie Stelle am Mutterprozess suchen. Natürlich kann dem Speicherverwalter auch mitgeteilt werden, an welcher Stelle im Speicher ein Block reserviert werden soll. Das ist wichtig, da der Reaper (s.o.) gezielt Prozesse entfernen können muss. Ist der Speicher voll, entfernt der Reaper solange Prozesse bis ein Block der gewünschten Größe frei ist.

3.2.4 Genetische Operatoren

Genetische Operatoren sind wie die Selektion Triebkraft der Evolution, sie ändern am Genmaterial vor. Ohne sie wäre eine Anpassung und dynamische Optimierung einer Population nicht möglich. Tierra kennt verschiedene genetische Operatoren: Mutation und Gen-Splicing-Operatoren. Diese Operatoren werden bei Geburt eines neuen Prozesses angewandt, und können auch in jedem Zeitpunkt des weiteren Lebens des Organismus angewendet werden. So werden die Mutationen simuliert die in der realen Welt z.B. durch kosmische Strahlen hervorgerufen werden.

3.2.5 Mutation

Zwei Arten der Mutation kommen in Tierra vor. Zum einen gibt es "Bit-Flips", d.h. ein einzelnes Bit einer 5-Bit-Anweisung kippt. Die andere Art der Mutation tauscht eine Anweisung, gegen eine zufällig ausgewählte aus dem Anweisungskatalog. Mutiert wird, wenn der Prozess geboren wird, wenn Code von einer Stelle zur anderen kopiert wird, oder zufällig irgendwann im Leben des Prozesses.

3.2.6 Gen-Splicing

Es gibt 3 Arten von Gen-Splicing in Tierra: Crossover, also die sexuelle Reproduktion, Insertion und Deletion. Jede Klasse dieser Operatoren kann auf 2 verschiedene Arten vorkommen: irgendwo im Genom oder an speziellen Punkten die durch Schablonen festgelegt wurden. Im biologischen Vorbild, der DNA, gibt es auch Stellen an denen, durch spezielle biochemische Eigenschaften, z.B. Insertion wahrscheinlicher ist.

Gen-Splicing-Operatoren werden auf einen Tochterprozess angewandt der geboren wird.

Crossover Ein Crossover in Tierra ähnelt einem 1-Punkt-Crossover bei genetischen Algorithmen. Ein Partner-Organismus wird zufällig ausgewählt, und im

gerade geborenen Tochterorganismus ein Crossover-Punkt bestimmt. Der kleinere Teil des Tochtergenoms wird dann durch den entsprechenden Teil des Partnergenoms ersetzt. Es ist Zufall ob sich das System entscheidet die Länge des Tochtergenoms dadurch zu verändern, oder beizubehalten.

Inserition Aus einem beliebigen Prozess im Speicher wird ein Codefragment beliebiger Grösse kopiert und an einer beliebigen Stelle des Tochtercodes eingefügt.

Deletion Eine Deletion löscht ein Stück aus dem Code eines Organismus. Zuerst wird die Länge des zu löschenden Teils bestimmt. Das Fragment kann bis zur Hälfte der Gesamtgenomlänge lang sein. Die Position des ersten zu löschenden Segments bestimmt, und die Deletion ausgeführt.

3.2.7 "Absichtliche" Fehler

Ab und an macht der virtuelle Prozessor "absichtliche" Fehler beim Ausführen einer Maschinencodeanweisung. Beispielsweise werden Vorzeichen geflipt, bei Addition um +/- 1 falsch gerechnet, um eine Stelle zuviel oder zuwenig geschiftet, beim Kopieren eine falsche Adresse (meist eine benachbarte) gewählt. "Absichtliche" Fehler (Flaws) sind keine genetischen Operatoren, da sie den genetischen Code nicht verändern. Sie wurden eingeführt um die biologische Unschärfe zu berücksichtigen, die in jedem Metabolismus vorkommen, wenn eine Reaktion nicht richtig abläuft, oder Nebenprodukte erzeugt.

3.2.8 Störungen

In bestimmten Intervallen entfernt das Betriebssystem einen Teil der Prozesse. Der Anwender kann das zeitliche Muster, und das Ausmass der Auslöschung bestimmen.

3.3 Arbeitsspeicher: die Suppe (Soup)

Ray nannte den Arbeitsspeicher seines digitalen Biotops "Suppe", angelehnt an den bekannten Begriff der Ursuppe. Der Arbeitsspeicher ist zirkulär aufgebaut, und wird dem Start der Simulation festgelegt. Die Grösse des Arbeitsspeichers kann während des Programmablaufs nicht verändert werden.

3.4 CPU

Tierra's Prozessor hat wie andere Prozessoren auch: einen Instruction Pointer, Register, einen Stack, einen Stack Pointer und Flags. Es gibt allerdings Besonderheiten, zum Beispiel den zirkulären Stack. Das bedeutet, dass nach mehrmaligen push der Stack nicht überläuft, sondern einfach die Daten am anderen Ende überschreibt. Dies ist notwendig um Fehler erst gar nicht entstehen zu lassen, wenn z.B. ein Prozess unverhältnismässig viel auf den Stack legen will, beschwert sich die Maschine zwar nicht, es bringt dem Prozess allerdings auch nichts. Später werden wir ähnliche Mechanismen kennenlernen, die auch zur Stabilität Tierra's beitragen sollen.

Weiters ist zu spezifizieren, dass Tierra in der Original-Version 4 Register hat, und der Stack 10 Words fasst.

3.5 Anweisungen

Beim Entwurf von verschiedenen Instruction-Sets für Tierra, wurden auf verschiedene Eigenschaften derselben Wert gelegt, um nah am biologischen Vorbild zu sein, und die Evolution nicht unnötig einzuschränken. Sonst kommen die Anweisungen realen Maschinencodeanweisung sehr nahe.

Es wurden verschiedene Instruction-Sets entwickelt. Man konnte beobachten, dass sich Unterschiede in der Qualität der Evolution zeigten. Dies bestätigte Rays Vermutung, dass Evolution stark von Medium in dem sie abläuft konditioniert wird.

Fehlerbehandlung Die Fehlerbehandlung sollte so gnädig wie möglich sein. Wäre sie so scharf wie realen Computer, hätte bereits eine Mutation an der falschen Stelle die gesamte Simulation lahmgelegt. So kann in Tierra keine Anweisung den virtuellen Rechner zum Absturz bringen. Bei schwerwiegenden Fehlern wird die Anweisung einfach übersprungen.

Minimierung Es sollte wenig Anweisungen geben. Deshalb sind alle Anweisung, nicht im klassischen Sinne, parametrisiert. Tierra verwendet also keine numerische Operanden. Nichtsdestotrotz kann Tierra in den Registern rechnen, bzw Zahlen erzeugen. Ein tierranische Anweisung besteht aus 5 Bit, also gibt es 32 verschiedene. Tierra besetzt alle 32 Möglichkeiten, und vermeidet so unausführbare Befehle schon per Definition.

Adressierung Da Tierra-Anweisungen nicht parametrisiert sind, können z.B. bei einem Sprung keine numerische Zieladresse angegeben werden. Tierra verwendet zu diesem Zweck Schablonen (Templates). Schablonen sind komplementäre Muster aus 1 und 0. 1011 ist Schablone zu 0100. Nun gibt es spezielle Anweisung zu nop0 und nop1 die Muster bzw. Schablonen darstellen können. Die Befehlssequenz

```
JMP;
NOP1;
NOPO;
NOP1;
NOP1;
```

sucht das nächste Vorkommen der Sequenz

```
NOPO
NOP1
NOPO
NOPO
```

und springt an diese Stelle. Ähnliche Vorgänge findet man in der Genetik. Die komplementäre Adressierung verhindert zudem, dass die Sequenz "sich selbst findet".

Syntax Auf Syntax wurde fast vollständig verzichtet. Alle Befehle sind atomar, d.h. sie brauchen keinen Kontext in dem sie ausgeführt werden müssen. Nur die Befehle, die oben beschriebenen Templates verwenden haben eine Syntax, da ja das Template spezifiziert werden muss.

3.6 Network Tierra

Es wird daran gearbeitet Tierra netzwerkfähig zu machen. Ähnlich wie das populäre Seti@home-Projekt, soll im Hintergrund die "Network Tierra"-Software laufen und ungenützte Rechenzeit verwenden. Rays Vision ist es ein grosses komplexes Reservat für digitale Organismen zu schaffen, die sich zwischen den Biotopen frei bewegen können. Das alles natürlichen nach dem Grundprinzip von Tierra immer in virtuellen Maschinen, um zu vermeiden dass sich eine Art Computervirus entwickelt. Den Forschern schwebt sogar vor auf diese Weise nützlichen Code zu finden, der dem Menschen bis jetzt verborgen blieb. Zur Zeit ist das "Network Tierra" in einer Testphase, mit Knoten an der ATR in Kioto, University of Delaware, Santa Fe Institute und der EPFL in Lausanne.

4 Organismen in Tierra

4.1 Der Vorfahre (Ancestor)

Thomas Ray impfte in seinem ersten Experiment mit Tierra einen Vorfahren in den Speicher Tierras. Dieser Vorfahre bestand aus 80 Instruktionen und hatte die Fähigkeit sich selbst zu kopieren.

Alle Organismen die sich später entwickelten, entstanden aus diesem Urganismus.

4.2 Parasiten

Nach einigen Millionen Instruktionen entstanden Organismen aus dem Urahn die kürzer waren als 80 Befehle. Diese Organismen konnten sich natürlich erfolgreicher fortpflanzen, da sie in der Rechenzeit die ihnen zugewiesen wurde, sich öfter kopieren konnten, als ihre längeren Verwandten. In Tierra ist eine Visualisierung implementiert, die anzeigt welche Länge Organismen haben, und wie viele Organismen einer bestimmten Länge gerade existieren. Nach weiterer Evolution passierte etwas bemerkenswertes. Es tauchten Organismen auf, die mit 45 Befehlen auskamen. Ray schätzte das ein Organismus mit funktionierender Selbstreplikationsschleife mindestens ein Länge von etwa 60 Instruktionen benötigt. Damit hatte er auch recht. Diese kleinen Organismen Organismen waren Parasiten. Verwendeten den Code von Wirtsorganismen um sich fortzupflanzen. An der Stelle wo sich die Schleife zum Kopieren befinden sollte, besass diese Organismen einen Sprungbefehl zur Schleife des Wirtes.

Es entwickelte sich ein evolutionäres Wettrüsten zwischen den Parasiten und den Wirtsorganismen. Die Wirte versuchten die Parasiten durch spezielle Mechanismen loszuwerden, worauf die Parasiten wiederum Mechanismen entwickelten um die Wirte auszutricksen. Ray war es also tatsächlich gelungen Evolution in einem digitalen Medium zu instanzieren.

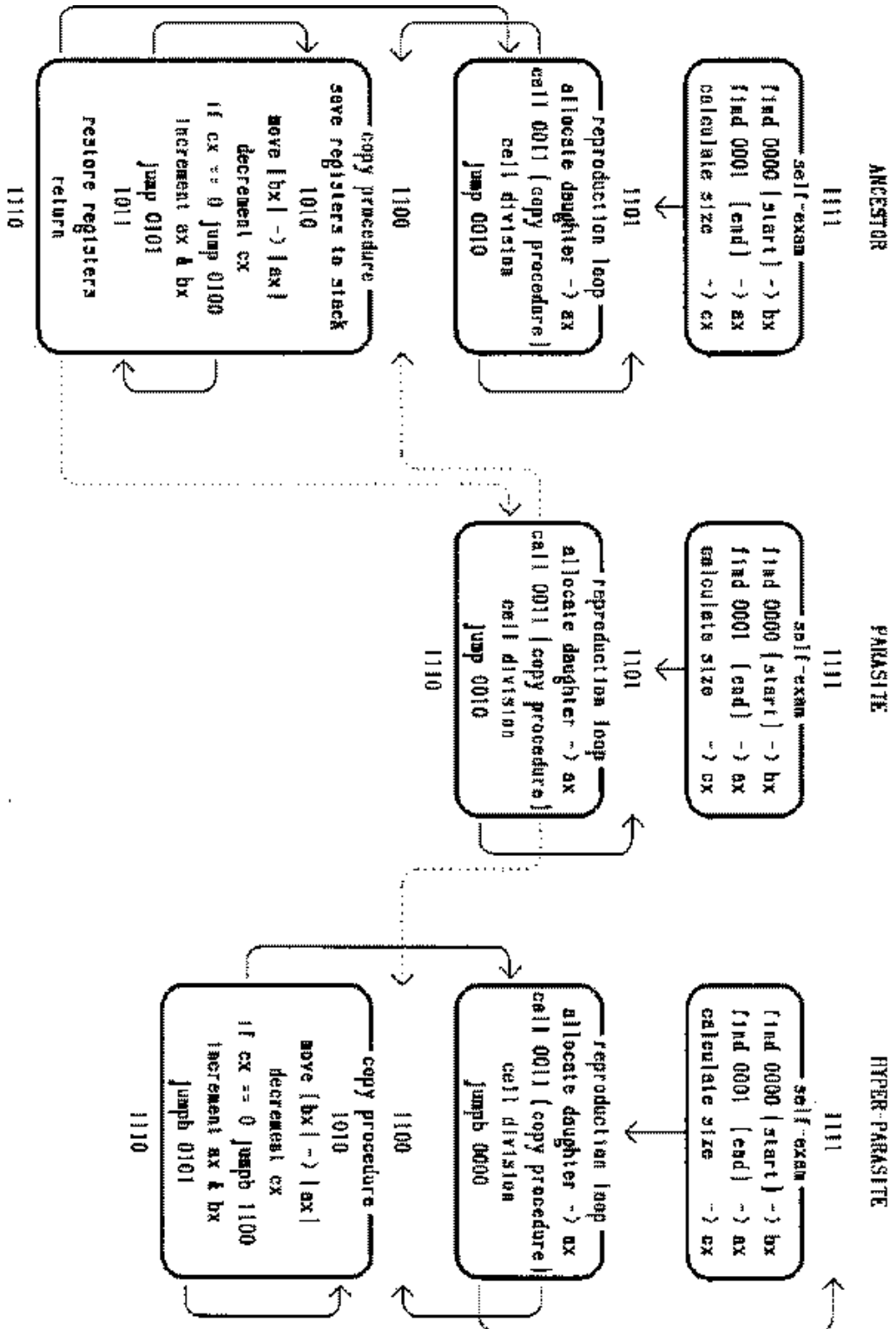


Figure 2: Vorfahre, Parasit und Hyperparasit

4.3 Hyperparasiten

Versucht ein Parasit einen Hyperparasiten zu befallen, wendet dieser einen Trick an. Er stiehlt die Rechenzeit des Parasiten, indem er in der Kopierschleife eine Rücksprunganweisung eingebaut hat, die in den Code des Hyperparasiten zurückspringt, und nicht in den Parasiten. Somit kann sich der Hyperparasit in der Rechenzeit des Parasiten reproduzieren. Die Hyperparasiten rotteten die Parasiten nach kurzer Zeit vollständig aus.

4.4 Soziale Hyperparasiten

Die soziale Hyperparasiten bildeten symbionitsche Gemeinschaften in dem sich die Anweisung zur Reproduktion teilten. So sparte sich jeder Organismus einige Befehle und konnte sich schneller reproduzieren. Diese Gemeinschaften zeigten sich als sehr stabil.

4.5 Hyper-Hyper-Parasiten

Diese Organismen nannte Ray auch Betrüger. Sie stellten sich zwischen zwei soziale Hyper-Parasiten, rissen den Istruktionszeiger an sich und verwendeten den Replikationscode der Hyperparasiten. Durch diese perfekte Anpassung an die Umgebung waren diese Organismen nur mehr 27 Instruktionen lang.

5 Multizelluläres, künstliches Leben in Tierra

Bis jetzt befassten wir uns nur mit simulierten Organismen die in der Natur am ehesten in Einzellern ihre Entsprechung finden. In der Natur setzten sich allerdings neben den Einzellen auch mehrzelligen Organismen durch, die die Aufgaben der Selbsterhaltung und Fortpflanzung, durch Verteilung auf viele einzelne Zellen erreichten. Durch die Erweiterung des Befehlssatzes und der CPU-Architektur ermöglichte T.S. Rays die Entstehung von multizellulären künstlichen Leben in Tierra.

5.1 Idee

Um das Konzept des multizelluläres Organismus von der organischen Form in die digitalen Form zu bringen, müssen wir zuerst verstehen welche Eigenschaften Multizellularität ausmachen:

- multizelluläre Lebewesen, sind anfangs einzellig und werden erst durch Zellteilung mehrzellig
- jede Zelle hat dasselbe genetische Material
- Differentiation: verschiedene Zellen können verschiedene Teile des Genoms exprimieren

In Tierra entspricht der Prozessor der Zelle und das Programm dem Genom. Daraus folgt, dass ein mehrzelliger Organismus in Tierra mehrere Prozessoren haben muss, die sich die Ausführung des Programmes aufteilen. Also ist in der multizellulären Implementation Tierras ein Parallelisierung der Organismen möglich. Die vorerst einzelliger (ein Prozessor) kann mittels einer Anweisung eine weitere Zelle (parallelen Prozessor) erzeugen. Jeder dieser Prozessoren

hat nicht nur Teil an der Rechenzeit, sondern bekommt genausoviel Rechenzeit wie der Ursprungsprozessor. Das heisst je mehr Prozessoren ein Organismus besitzt, desto mehr Instruktionen kann er ausführen. Alle Zellen eines Organismus haben offensichtlich dasselbe genetische Material da es nur einmal pro Organismus vorkommt.

5.2 1. Implementation

Zur Einführung von Parallelität in Tierra muss das Instructionset der Maschine erweitert werden..

SPLIT hat zur Folge dass aus einem Prozessor zwei neue parallele Prozessoren erzeugt werden. Wie ein neuer Prozessor erzeugt werden zunächst die register, der Stack und der Instruktion-Pointer des Prozessors kopiert, von dem der SPLIT-Befehl ausgeführt wird. Um nach der Teilung zwischen den Prozessoren unterscheiden zu können, wird im DX-Register eine eindeutige Selbst-Adresse gespeichert. Die Inhalt der DX-Register wird bei einer Teilung um ein Bit nach links geschoben, und beim neuen Prozessor wird das niederwertige Bit auf 1 gesetzt. Wird in einem Program eine zweite Splitanweisung ausgeführt werden beide Prozessoren geteilt es entstehen also 4 Prozessoren. Ein Organismus kann maximal 16 Prozessoren haben.

JOIN ist ein weiterer Befehl der zur Parallellisierung nützlich ist. Führt ein paralleler Prozessor JOIN aus wartet er solange bis alle anderen Prozessoren auch ein JOIN ausgeführt haben. Danach werden alle parallelen Prozessoren wieder zum ursprünglichen Prozessor vereint. Tierra ist limitiert darauf, nur einen Prozessor zuzulassen, um die Teilung von Mutter und Tochter durchzuführen. Deshalb wurde ein JOIN-Befehl direkt vor der Ausführung der Teilung eingeführt. Dieser Befehl hat keine Entsprechung in der Natur.

5.2.1 Ergebnisse

Wie im ursprünglichen Tierra wurde ein Vorfahre in den Speicher kopiert. Dieser war fast identisch mit dem originalen Vorfahren. Er unterschied sich nur darin, dass die Kopierschleife parallelisiert war: zwei CPUs kopierten jeweils die Hälfte des Genoms, einer die geraden und einer die ungeraden Adressen. Der Prozess der Parallellisierung zeigte sich als ausserordentlich schwierig, da der ursprünglich Befehlssatz nicht gerade geschickt darin war auf Registern zu operieren. Dies hatte Auswirkung auf die Evolution, da neue Organismen erstmal die komplizierte Parallellisierung schaffen mussten um aus Mehrzelligkeit nutzen zu ziehen. Die Replikationsgeschwindigkeit verdoppelte sich beinahe, da ja nun 2 Prozessoren arbeiteten. Die Replikationszeit wurde zum neuen Indikator für Erfolg in Tierra, vorher war es die Länge eines Organismus gewesen. Da jetzt mehrere Prozessor erlaubt waren, konnte sich ein langer Organismus mit vielen Prozessoren schneller fortpflanzen, als ein kurzes Genom mit wenigen Prozessoren. In dieser Implementation allerdings, entwickelten sich keine Organismen die mehr als 2 Prozessoren in Anspruch nahmen.

5.3 2. Implementation

Da die “evolutionäre Ausbeute” in der ersten Implementation eher schwach war, entschieden sich Kurt Thearling und Thomas Ray den Befehlssatz weiter zu erweitern.

ZeroD setzt das DX-Register auf 0.

SHR ist ein Schiebepfehl, der den Inhalt des des CX-Register um 1 Bit nach rechts schiebt, also ein ganzzahlige Division durch 2 durchführt. Dies ist interessant, weil das CX-Register meistens die Länge des Organismus enthält. Wird nach einer SPLIT-Anweisung diese Instruktion ausgeführt, enthält das CX die Portion des Genoms die ein paralleler Prozessor noch kopieren muss.

offAACD ist eine Offset-Anweisung. D.h. sie nimmt die eindeutige ID des Prozessors (gespeichert in DX) und multipliziert sie mit der Portion die dieser Prozessor zu kopieren hat (gespeichert in CX), und addiert sie zur Speicheradresse an der der Organismus beginnt (gespeichert in AX). Dies ist sehr nützlich zur Parallellisierung der Kopierschleife, da sie es erleichtert das Genom, wie eine Wurst zu portionieren.

offBBCD macht fast das selbe wie offAABC, nur dass sie das Ergebnis der Multiplikation zum BX-Register addiert.

5.3.1 Ergebnisse

Ein neuer Urahn wurde erschaffen, der die neuen Befehle ausnutzte. Er verwendete 2 Prozessoren, wieder kopierte jeder eine Hälfte des Genoms. Voerst zeigten sich diesselben Phänomen wie in anderen Tierra-Experimenten: Parasiten, Hyperparasiten und Resistenzbildung. Nach 215 Millionen Zeittakten gab es jedoch eines Evolutionssprung. Organismen lernten mit 4 Prozessoren umzugehen. Die Organismen die 2 Prozessoren verwendeten waren 44 Anweisungen lang, die 4-Prozessor-Organismen 52 Anweisungen. Da jeder Prozessor aber weniger zu kopieren hatte ($52/4 = 13 < 44/2 = 21$), erreichten diese Organismen die Überhand. Nach weiteren Generationen entwickelten sich auch Organismen mit 4-Prozessoren der der Länge 40 (genannt 40aba), 44, 48. Als man die Organismen untersuchte stellte sich heraus, dass sie sich im Algorithmus nicht unterschieden. Die unterschiedliche Länge resultierte aus künstlichen Introns, also Befehlen die keine Wirkung hatten. Die Sinn dieser Befehle bestand darin die Länge des Organismus auf ein Vielfaches von 4 zu bringen, um ihn gleichmässig auf alle Prozessoren verteilen zu können. Das ist der Organismus 00040aba:

```
Nop0           ; beginning template
adrb           ; find beginning + template size
nop1           ;
subAC          ; sub template size from beginning
movAB          ; put beginning in BX
adrf           ; find end
nop0           ;
nop0           ;
```

```

subCAB      ; calculate size
mal         ; allocate space for daughter
incC        ; intron (since CX no longer used)
split       ; 2 CPUs
ifz         ; intron (since CX cant be zero)
movCD       ; intron (since ifz not true)
shr         ; size = size / 2
offAACD     ; split genome into 2 halves &
offBBCD     ; adjust AX and BX accordingly
zeroD       ; zero out DX before second split
pushB       ; save beginning on stack
shr         ; size = size / 2
split       ; 4 CPUs
offAACD     ; re-partition genome into 4 and
offBBCD     ; adjust AX and BX accordingly
nop1        ; copy loop starts here
nop0        ;
movii       ; copy instr from mother to daughter
decC        ; decrement number of instr to copy
ifz         ; if number of instr to copy == 0
jmp         ; jump back to just before join
nop0        ;
nop1        ;
join        ; join up multiple CPUs
divide      ; divide mother and daughter
ret         ; return to beginning of creature
nop1        ; ending template
nop1        ; ending template

```

6 Fazit und Ausblick

Tierra ermöglicht es Forschern erstmals einen genauen Einblick in Mechanismen der Evolution zu geben. Es wurde gezeigt, dass es möglich ist Instanzen von Evolution zu schaffen, und somit wurde bewiesen, dass es sich dabei um kein Phänomen handelt, das nur in unsrer Natur vorkommen kann. Seit dessen Entwicklung beschäftigen sich viele Interessierte sowohl mit den Ergebnissen, als auch mit der Weiterentwicklung des Programms. Es kursieren sehr viele Tierra-ähnliche Programme, die verschiedene Aspekte der Evolution beleuchten.

7 Anhang

7.1 Befehlssatz von Tierra

```

nop_0      (no operation 0; used in templates)
nop_1      (no operation 1; used in templates)
orl        (flip the low bit of cx)
shl        (shift left cx)
zero       (zero cx)
if_cz      (if cx=0 do next instruction)

```

sub_ab	(subtract bx from ax, result into cx)
sub_ac	(subtract cx from ax, result into ax)
inc_a	(increment ax)
inc_b	(increment bx)
dec_c	(decrement cx)
inc_c	(increment cx)
push_ax	(push ax onto the stack)
push_bx	(push bx onto the stack)
push_cx	(push cx onto the stack)
push_dx	(push dx onto the stack)
pop_ax	(pop top of stack into ax)
pop_bx	(pop top of stack into bx)
pop_cx	(pop top of stack into cx)
pop_dx	(pop top of stack into dx)
jmp	(move ip to template)
jumpb	(move ip backward to template)
call	(call a procedure)
ret	(return from procedure)
mov_cd	(move cx to dx)
mov_ab	(move ax to bx)
mov_iab	(move instruction pointed to by bx to address in ax)
dr	(address of nearest template into ax)
adrb	(search backwards for template)
adrf	(search forwards for template)
mal	(allocate memory for daughter cell)
divide	(cell division)

7.2 Quellen

[RayTS1996] Ray, T. S. 1996. Netlife - das Schaffen eines Dschungels im Internet. Stefan Iglhaut, Armin Medosch, Florian Rotzer (eds.), Stadt am Netz, Ansichten von Telepolis. Pp. 118-126.

[RayTS1999] Ray, T. S. 1999. An Evolutionary Approach to Synthetic Biology: Zen and the Art of Creating Life. In: Virtual Worlds: Synthetic Universes, Digital Life and Complexity, Jean-Claude Heudin [ed.], 29-65. New England Complex Systems Institute Series on Complexity, Perseus Books (Sd), Pp. 320.

[RayTS2001] Ray, T. S. 2001. Overview of Tierra at ATR. In: "Technical Information, No.15, Technologies for Software Evolutionary Systems". ATR-HIP. Kyoto, Japan.

[RayTS1999] Ray, T. S., and Hart, Joseph. 1999. Tierra Tutorial. 1999 Genetic and Evolutionary Computation Conference Tutorial Program, 367-394. Morgan Kaufmann, San Francisco. Presented at GECCO (Genetic and Evolutionary Computation Conference)

[LevyS1993] Steven, Levy 1993. Künstliches Leben aus dem Computer. Droemer & Knauer Jahr: 1996 OBr. 460 S. Erschienen als Knauer Taschenbuch Seite 269 - 290.