Mathware & Soft Computing 8 (2001), 137-152

A Formal Analysis of the Computational Dynamics in **GIGANTEC**

A. Badr
Dept. of Computer Science
Faculty of Computers & Information. Cairo University
e-mail: ruaab@rusys.EG.net

Abstract

An evolutionary algorithm formalism has been forwarded in a previous research, and implemented in the system **GIGANTEC**: Genetic Induction for General Analytical Non-numeric Task Evolution Compiler [Bad98][Bad99]. A dynamical model is developed to analyze the behaviour of the algorithm. The model is dependent in its analysis on classical Compilers Theory, Game Theory and Markov Chains and its convergence characteristics. The results conclude that a limiting state is reached, which is independent of the initial population and the mutation rate, but dependent on the cardinality of the alphabet of the driving L-system.

Keywords. Genetic Algorithms- Evolutionary Algorithms- Finite State Machines- Petri Nets- Symbolic Computing- Model Design- State Spaces- Mutation- Search- Exploration- Stochastic Context Sensitive Grammar- Stochastic L-system- Computational Dynamics- Statistical Model- Markov Chains.

1 Introduction

The **GIGANTEC** formalism is a general framework suited for linguistic large-scale designs. The problem considered is the evolution of symbolic design plans. A methodology of design is forwarded. The design should provide a 'context sensitive L-system' that defines the behavioural interaction between abstract system tasks. There are three levels of abstraction, the finite state machine (FSM), the petri nets and component automata. The tokens generated by the FSMs abstract layer are forwarded into a petri net dispatcher (PND). The PND in turn distributes and selects

the potential sub-petri net to fine tune the abstract task represented by that token [Bad98][Bad99].

Formal Dynamics of the system reveal a close relationship to classical Compilers Theory, Statistics, and Game Theory. L-system grammars are provided for abstract system processes. Several operators are defined on these grammars for the purpose of generating a framework for parsing actions to be taken; similar to the operators defined in parsing theory. Finally, a Deterministic Finite Automaton (DFA) template is generated. Game theoretic models usually employ techniques such as bi-matrices to define strategies which each player abides by. A similar analysis is carried out for **GIGANTEC**, especially through the competition between population individuals, in which the process of selection determines the fittest to survive. An evolutionary stable system is reached known as the *Nash Equilibrium Pair*.

A Markov Chain analysis is presented for mutation in the Evolutionary Algorithm **GIGANTEC** [Bad98][Bad99]. The difficulty of such analysis arises from the fact that the Evolutionary Algorithm is too coarse. That's to say, too many parameters are involved and thus, many simplifying assumptions are made. The analysis tries to model the algorithm as a whole, rather than only components of the algorithm. A complete model is more predictive than a partial one. Mutation is carried only on the layer of the abstract brain FSMs. This is the layer involved in the Markovian analysis. Petri nets are processing elements and will not be involved in the markovian analysis presented.

2 Problem Definition

Mathematically speaking, the problem can be formulated as follows: an intelligent design, represented algorithmically by a finite state machine, is intended to complete a given task:

$$T = \{ \Lambda_1, \Lambda_2, ..., \Lambda_m \}$$

where $\Lambda_1, ..., \Lambda_m$ are subtasks which compose T. Each subtask is represented by a Petri net and is defined by a set of performance specifications to be fulfilled by that subtask:

$$\chi = [x_1, x_2, ..., x_n]$$

The state variables describing the current status of an intelligent design are given by:

$$S_{\Lambda_i} = \{ s_{i1}, s_{i2}, ..., s_{in} \}$$

where $\Lambda_i = \Lambda_1, \Lambda_2, ..., \Lambda_m$

S_{Λ_i} = states for task Λ_i

and s_{ij} indicates the performance specification for the state variable x_j in the subtask t_i such that $x_j \subset s_{ij}$ indicates that specification {ij} has been met. A

potential function describes the cost of task T in terms of its subtasks, and can be defined as:

$$pot(T,x) = v_T$$

where $$v_T = \Psi(v_{\Lambda_i}) = \Psi[\sum_{i=1}^{m} pot(\Lambda i, s_{\Lambda_i})]$$

and Ψ is a problem specific function which is usually a composite one and evaluated according to conflict matrices, cubes or hypercubes used to determine the fittest finite state machine on basis of a conflict between FSM_i and FSM_j among the population. This will be clarified in later sections and in [Bad98][Bad99].

The intelligent design has a set of available low-level designs:

$$D = \{ D_1, D_2, ..., D_r \}$$

where *r* is the population size and each design D_i is composed of sub-designs d_{ij} associated with the m subtasks of T:

$$D_i = \{ d_{i1}, d_{i2}, ..., d_{im} \}$$

Each D_i is represented by a finite state machine. Each d_{ij} is represented by a Petri net or a Petri net component automaton. The intelligent design problem is defined as the intelligent selection of the optimum design D_{opt} from D given a specified task T. The optimal design D_{opt} is defined as the design that maximizes the 'potential' of the system, and the probability that the task T is completed successfully, such that:

$$Pr(T) = pot(T,x) = \text{maximum}$$

where Pr(T) is the probability of a selected design to complete the required task. As mentioned, this probability is determined by a problem-specific function Ψ which is in turn evaluated according to conflict matrices.

The design itself, made up of m components, is represented mathematically as follows:

$$D_i = FSM_i \xrightarrow{[Token(s)]} \begin{bmatrix} f_1(\overline{\mu}_1, \overline{\lambda}_1, PN_1, CA_1) \\ \vdots \\ f_j(\overline{\mu}_j, \overline{\lambda}_j, PN_j, CA_j) \\ \vdots \\ f_m(\overline{\mu}_m, \overline{\lambda}_m, PN_m, CA_m) \end{bmatrix}$$

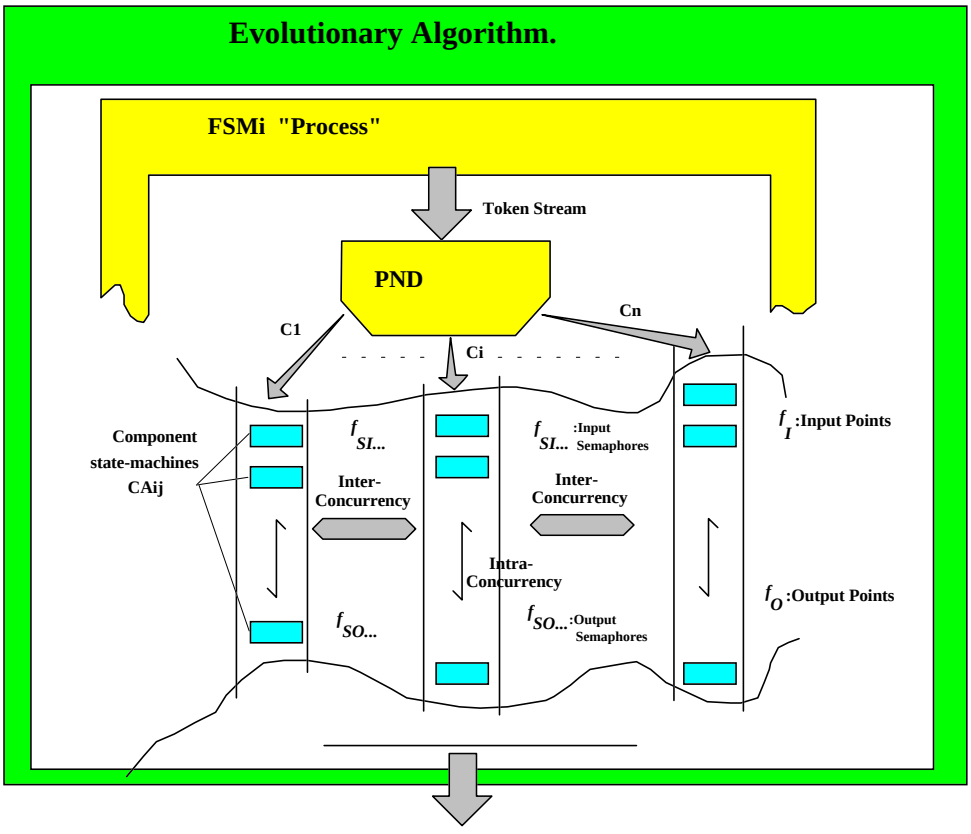
where $\overline{\mu}_j$ and $\overline{\lambda}_j$ are the input and output points of the *j*th component sub-design (the *j*th Petri net) referred to as 'semaphores' in the formal description of the representation data structures [Bad98][Bad99]. The CA_j are the 'component automata' of PN_j.

3 The Evolutionary Algorithm

3.1 Schematic Model

The evolutionary algorithm's representation/ operators in its executable version functions as follows. The outer layer is the global evolutionary algorithm which acts upon the *abstract brain layer* (FSMs) by problem specific evolutionary operators. The abstract brain layer generates 'tokens' to be received by the inner layer of 'decomposable' petri nets. Each token has a specific petri net to receive it. The token represents an 'abstract entity' (a high-level ADT) that identifies the respective Petri net. This specific petri net act as a simulator of the abstract task's functional behaviour. The petri nets are decomposable in the sense that they can be partitioned into *component automata* [Bat93]. Component automata represent the inner-most 'discrete' and 'non-decomposable' task.

The schematic decomposition of the problem-*dependent* part is shown in the following diagram:



3.2 L-system and Context Sensitive Grammar

Mutation operators in the system are guided by an L-system. Each rule in the L-system is of the form:

$$< \Lambda_L >< \Lambda_M >< \Lambda_R >\rightarrow \Lambda_0 \Lambda_1 \cdots \Lambda_n$$

where:

Λ_R : Right context of Λ_M . i.e. a process or task that should be performed after Λ_M .

Λ_L : Left context of Λ_M . i.e. a process or task that should be performed before Λ_M .

Λ_i : A set of tasks or processes that constitute Λ_M .

3.2.1 Formal Dynamics

To clearly understand the dynamics of the L-system and the effects of the operation of the Evolutionary Algorithm, several operators must be defined. These operators are borrowed from classical Compilers theory. [Aho86][Lou97].

Definition. The Dot Operator

The dot operator defines an item of a grammar **G**, which is a production of **G** with a dot at some position at the right side. Thus the production:

$$< \Lambda_L >< \Lambda_M >< \Lambda_R >\rightarrow \Lambda_0 \Lambda_1 \cdots \Lambda_n$$

yields the n+2 items as follows:

$$< \Lambda_L >< \Lambda_M >< \Lambda_R >\rightarrow \bullet \Lambda_0 \Lambda_1 \cdots \Lambda_n$$

$$< \Lambda_L >< \Lambda_M >< \Lambda_R >\rightarrow \Lambda_0 \bullet \Lambda_1 \cdots \Lambda_n$$

$$\vdots$$

$$< \Lambda_L >< \Lambda_M >< \Lambda_R >\rightarrow \Lambda_0 \Lambda_1 \cdots \bullet \Lambda_n$$

Definition. The Closure Operator.

If **I** is a set of items for a grammar **G**, the *closure(I)* is the set of items constructed from **I** by the following algorithm:

function *closure (I):*
begin
  J:= I;
  repeat
    for *each item* $< L >< M >< R >\rightarrow \alpha \bullet B \beta$ *in J*
    and each production $< L_B >< B >< R_B >\rightarrow \gamma$ *of* **G**
    such that $< L_B >< B >< R_B >\rightarrow \bullet \gamma$ *is not in J* ***do***
    add $< L_B >< B >< R_B >\rightarrow \bullet \gamma$ *to J;*
  until *no more items can be added to J;*
  return *J*
end [Aho86]

Definition. The Goto Operation.
If **I** is a set of items and **X** is a grammar symbol, then goto(I,X) is defined to be the closure of the set of all items $< L >< M >< R >\rightarrow \alpha X \bullet \beta$ such that $< L >< M >< R >\rightarrow \alpha \bullet X\beta$ is in **I**, where each Greek letter represents a collection of non-expandable discrete tasks or processes, and capital English letters represent expandable tasks.

Definition. The Augmented Grammar.
If **G** is a grammar with start symbol (starting task) S, then G'**,** the augmented grammar for **G** is **G** with a new start symbol S' and production $S' \rightarrow S$.

Definition. The Canonical System.
The algorithm to generate the canonical system **C** of sets of L-system items for an augmented grammar G' is shown:

procedure *generate-canonical-system(*G'*);*
begin
C:={closure({[$S' \rightarrow \bullet S$*]})};*
repeat
for *each set of items I in C and each grammar symbol X such that goto(I,X) is not empty and not in C* ***do***
add *goto(I,X) to C*
until *no more sets of items can be added to C*
end

for example, given the following L-system:

$$< \Lambda_{L1} >< \Lambda_{M1} >< \Lambda_{R1} >\rightarrow \Lambda_{11}\Lambda_{12}\cdots\Lambda_{1n}$$
$$< \Lambda_{L2} >< \Lambda_{M2} >< \Lambda_{R2} >\rightarrow \Lambda_{21}\Lambda_{22}\cdots\Lambda_{2n}$$
$$\vdots$$
$$< \Lambda_{Lm} >< \Lambda_{Mm} >< \Lambda_{Rm} >\rightarrow \Lambda_{m1}\Lambda_{m2}\cdots\Lambda_{mn}$$

where any entry Λ_{ij} can be empty, then the augmented L-system is as follows:

$$\Lambda'_s \rightarrow \Lambda_{M1}$$
$$< \Lambda_{L1} >< \Lambda_{M1} >< \Lambda_{R1} >\rightarrow \Lambda_{11}\Lambda_{12}\cdots\Lambda_{1n}$$
$$< \Lambda_{L2} >< \Lambda_{M2} >< \Lambda_{R2} >\rightarrow \Lambda_{21}\Lambda_{22}\cdots\Lambda_{2n}$$
$$\vdots$$
$$< \Lambda_{Lm} >< \Lambda_{Mm} >< \Lambda_{Rm} >\rightarrow \Lambda_{m1}\Lambda_{m2}\cdots\Lambda_{mn}$$

and the canonical system for the augmented L-system is as follows:

I_0 :
$\Lambda'_s \rightarrow \bullet \Lambda_{M1}$
$< \Lambda_{L1} >< \Lambda_{M1} >< \Lambda_{R1} > \rightarrow \bullet \Lambda_{11}\Lambda_{12}\cdots\Lambda_{1n}$
$< \Lambda_{L2} >< \Lambda_{M2} >< \Lambda_{R2} > \rightarrow \bullet \Lambda_{21}\Lambda_{22}\cdots\Lambda_{2n}$
$\vdots$
$< \Lambda_{Lm} >< \Lambda_{Mm} >< \Lambda_{Rm} > \rightarrow \bullet \Lambda_{m1}\Lambda_{m2}\cdots\Lambda_{mn}$

I_1: goto(I_0 , Λ_{M1})
$\Lambda'_s \rightarrow \Lambda_{M1} \bullet$

I_2 : goto(I_0 , Λ_{11})
$< \Lambda_{L1} >< \Lambda_{M1} >< \Lambda_{R1} > \rightarrow \Lambda_{11} \bullet \Lambda_{12}\cdots\Lambda_{1n}$

I_3 : goto(I_0 , Λ_{21})
$< \Lambda_{L2} >< \Lambda_{M2} >< \Lambda_{R2} > \rightarrow \Lambda_{21} \bullet \Lambda_{22}\cdots\Lambda_{2n}$

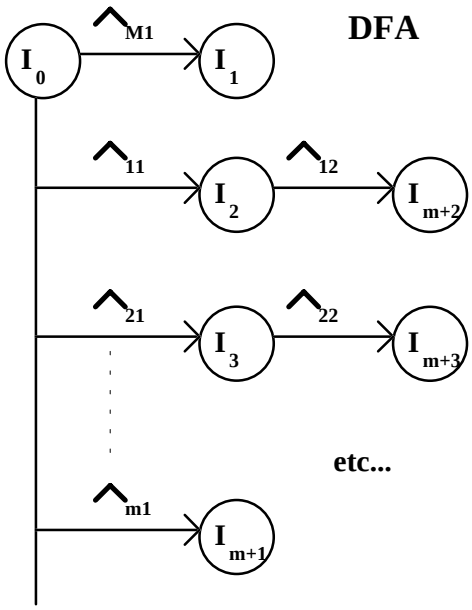
$\vdots$
I_{m+1} : goto($\mathbf{I}_0$, Λ_{m1})
$< \Lambda_{Lm} >< \Lambda_{Mm} >< \Lambda_{Rm} > \rightarrow \Lambda_{m1} \bullet \Lambda_{m2}\cdots\Lambda_{mn}$

I_{m+2} : goto(I_2 , Λ_{12})
$< \Lambda_{L1} >< \Lambda_{M1} >< \Lambda_{R1} > \rightarrow \Lambda_{11}\Lambda_{12} \bullet \Lambda_{13}\cdots\Lambda_{1n}$

I_{m+3} : goto(I_3 , Λ_{22})
$< \Lambda_{L2} >< \Lambda_{M2} >< \Lambda_{R2} > \rightarrow \Lambda_{21}\Lambda_{22} \bullet \Lambda_{23}\cdots\Lambda_{2n}$

$\vdots$

and so on....
Thus, the DFA (Deterministic Finite Automaton) for the viable prefixes in the canonical system:



3.3 System Evaluation

A particular design representation is evaluated by carrying out a *conflict* process (competition) between an FSM*i* and FSM*j* ($j \neq i$ and *j:1..population-size*). Conflict matrices, cubes or hypercubes are constructed to provide conflict-indices to be accumulated, as shown in the algorithm EvalFSM below:

Algorithm. EvalFSM.

```
begin
    for i = 1 to population_size
    begin
        for j = 1 to population_size
        begin
            -check i not equal to j, if YES continue, else break to
next iteration.
            - run conflict between FSMi and FSMj
            begin
                for n = 1 to contest_length
                    fitness( FSMi ) = fitness (FSMi) +
                            conflict_index [i][j];
            end
        end
    end
end
Scan population i: 1 -> population_size for FSMk, the fittest
individual;
```

The concept of the conflict-index is borrowed from game theory. When applied in fuzzy terms, it is analogous to fuzzy associative memories (FAMs).

An example of a conflict matrix:

			FSMi				
			Variable k				
		....	L	M	H	....	
FSMj	Variable k	L					
		M		Conflict-index			
		H					
		....					

3.4 The Algorithm

The proposed algorithm is shown, in an abstract form, as follows:

Algorithm PEA: Proposed Evolutionary Algorithm.
begin
***initialize population** $P(t)$ **- by applying L-system rules and Add-node mutation operator.**
***for gen = 1 to generation_size do**
begin
- Evaluate $P(t)$ **by competition between FSMs (EvalFSM algorithm).**
- Sort population $P(t) \xrightarrow{sort} P'(t)$
-for i = 1 to population_size do
begin
- Apply roulette (SUS) to $P'(t)$ **to select individual** η_i.
- Mutate η_i **- according to mutation operators' probabilities.**
end
-Pass the new population $P''(t)$ **to next generation** $P(t) = P''(t)$**;**
end
end.

As shown, the PEA utilizes components such as the EvalFSM algorithm, for evaluating fitnesses, the Stochastic Universal Sampling (SUS) [Bak87] algorithm with elitist strategy and the five mutation operators discussed in [Bad98][Bad99].

4 A Formal Analysis of Dynamics

The dynamics of the proposed EA are largely dependent on the *game* theoretic models. The conflicts can be 'symmetric' in the sense that all contestants are in the same situation and strength; however, many conflicts are asymmetric- where the population individuals/ species compete for the available resources. In fact, asymmetries are not only incidental, but are essential features of *Strategy Evolution.* For simplicity, the analysis will be carried out for conflicts settled in pair-wise encounters and finite numbers of pure strategies- a concept similar to 'bimatrix' games [Hof88].

Let $E_1, E_2, ..., E_n$ denote the phenotype of population *X*, and $F_1, F_2, ..., F_m$ those of population *Y*, with frequencies $x_1, x_2, ..., x_n$ and $y_1, y_2, ..., y_m$ respectively. By x_i we denote the frequency of E_i, and by y_j that of F_j. Hence, the state of population *X* is given by a point **x** in S_n (space of *E*) and that of population *Y* by a point **y** in S_m (space of *F*). If an E_i-individual is matched against an F_j-individual, the payoff will be a_{ij} for the former, and b_{ji} for the latter. Thus, the conflict is described by two payoff matrices $A = [a_{ij}]$ and $B = [b_{ji}]$. The expressions $\mathbf{x}.Ay$ and $\mathbf{y}.Bx$ are the average payoffs for population *X* and population *Y*, respectively. In an alternative interpretation, $E_1, E_2, ..., E_n$ would correspond to pure strategies of contestant **x**, and $F_1, F_2, ..., F_m$ to the pure strategies for contestant **y**. Points $x \in S_n$ and $y \in S_m$ correspond to mixed strategies and $\mathbf{x}.Ay$ would be the expected payoff for the **x**-strategy against the **y**-strategy.

Evolutionary stability is considerably more restricted for asymmetric than symmetric conflicts. That's to say, if the phenotype E_i is stable in contests against the population *Y*. It must do at least as well, then, as all competing phenotypes, i.e.

$$(Ay)_i \geq (Ay)_k, \forall k \neq i$$

Thus, a definition can be formulated. A pair of strategies *(p,q)* with $p \in S_n$ and $q \in S_m$ is said to be evolutionary stable if:

$$p.Aq > x.Aq, \forall x \in S_n, x \neq p$$

and $q.Bp > y.Bp, \forall y \in S_m, y \neq q$

This leads to a well known definition in classical game theory known as the *Nash Equilibrium.*

Definition. ***Nash Equilibrium Pair.*** [Hof88], [Ioo90], [Bar91]
A pair of strategies $(p,q) \in S_n \times S_m$ is called a Nash Equilibrium Pair if

$$p.Aq \geq x.Aq, \forall x \in S_n$$
$$\text{and } q.Bp \geq y.Bp, \forall y \in S_m$$

The standard evolutionary dynamics for this type of contests give the following differential equation on $S_n \times S_m$:

$$\dot{x}_i = x_i[(Ay)_i - x.Ay]$$
$$\text{and } \dot{y}_j = y_j[(Bx)_j - y.Bx]$$

A slightly different dynamics include normalization by mean payoff:

$$\dot{x}_i = x_i \frac{[(Ay)_i - x.Ay]}{x.Ay}$$

$$\text{and } \dot{y}_j = y_j \frac{[(Bx)_j - y.Bx]}{y.Bx}$$

These equations are motivated by the discrete time model, which is based on the assumption that the frequency of the E_i contestant in the next generation, x'_i, is proportional to $x_i(Ax)_i$, since $(Ax)_i$ is the mean payoff for E_i. In a population of FSMs, where each FSM complexity is taken into consideration, FSMi (equivalent to E_i) competes against FSMj (equivalent to F_j), with complexities CE_i and CF_j, where the complexity can be measured in terms of its number of states.

5 Selection Operator: Analysis of Dynamics

Multi-modality of the fitness function, which allows different species (namely, sub populations corresponding to local maxima/minima) to co-exist, is taken into account in the system GIGANTEC. Let η_j be a generic individual (a FSM description of a design policy) belonging to the population *P(t)*. The population,

$$P(t) = \{\eta_1^{x_1(t)}, \ldots, \eta_m^{x_m(t)}\}$$

is a multi-set, containing *m* different formulas ($1 \leq j \leq m$), each has the same complexity and multiplicity $x_j(t)$. Let $COV(\eta_j)$ be a subset of T (Tasks) covered by η_j then,

$$COV(P(t)) = \bigcup_{j=1}^{m} COV(\eta_j) \quad with \quad |COV(P(t))| = S$$

In this model, the set $P'(t)$ is selected with replacement from *P(t)* by assigning to each η_j $(1 \leq j \leq m)$ a probability proportional to its total fitness,

$$P_j = f_j\, x_j(t) \Big/ \sum_{i=1}^{m} f_i\, x_i(t) \quad (1 \le j \le m)$$

The fitness $f_j = f(\eta_j)$ should take into account completeness, consistency and simplicity of the FSM η_j.

The fitness evaluation mechanism is dependent on competition between each FSM (η_j) and the rest of FSMs ($\eta_i, 1 \le i \le m, i \ne j$). Thus the evolution of an 'ideal average population' *P(t)* is defined in such a way that the number of copies of each η_j in *P(t)* is equal to the mean value $\bar{x}_j(t)$ of the stochastic variable $x_j(t)$. For simplicity (and reality of competition which is carried out in conflicts of individuals of cardinality = 2), the case of *P(t)* containing only two individuals, namely FSMs η_1 and η_2 will be considered in the analysis. This simplification does not change the nature of the problem, but allows the results to be obtained in closed form. Considering the case of 'generational replacement' (whole population is replaced), let $x(t) = x_t$ be the number of copies of η_1 in *P(t)* and $c = |P'(t)|$, then $(c - x_t)$ that of η_2. Then the probability of η_1 to be selected in one trial is given by,

$$P_t = \frac{f_1 x_t}{f_1 x_t + (c - x_t) f_2}$$

The probability distribution of the number of copies of *x(t+1)* of η_1 in $P'(t)$, given that there are *x(t)* copies in *P(t),* will be given by a binomial distribution with c trials probability of success P_t.

The mean value of x_{t+1} is $\bar{x}_{t+1} = c\, P_t$; obviously, $\bar{x}_{t+1}$ depends on x_t which is hidden inside P_t. Thus, considering the mean value $\bar{x}_{t+1}$ corresponding to the ideal population, we obtain the following recurrent equation:

$$x_{t+1} = \frac{\lambda . x_t}{c + (\lambda - 1) x_t}$$

where $\lambda = f_1 / f_2$.

The solution for this relation is then,

$$x(t) = \frac{c . x_o . \lambda^t}{c - x_o . (1 - \lambda^t)} \quad \textit{where} \quad x_o = x(0)$$

Thus, the asymptotic behaviour of *x(t)* in *P(t),*

$$\lim_{t \to \infty} \bar{x}(t) = \begin{cases} M & \textit{if} \quad \lambda > 1 \\ x_o & \textit{if} \quad \lambda = 1 \\ 0 & \textit{if} \quad \lambda < 1 \end{cases}$$

These limits tell us that the best individual (FSM disjunct), will, in the average, eventually take over the whole population and kill the other one. A similar analysis can be found for other selection operators, such as the 'Universal Suffrage Operator' and 'Sharing function operator' in [Gio94].

6 Mutation Operators: A Markov Chain Analysis

The expected time evolution of individuals in a population undergoing selection and mutation, requires a model of C^X simultaneous equations where X is the average complexity of the individual (Number of states in a FSM) and C is the cardinality of the alphabet of the L-system provided. Let FSM_i and FSM_j be arbitrary FSMs of complexity X. Let the proportion of a FSM FSM_j at time t be denoted as $p^{(t)}_{FSM_j}$. Then the expected time evolution of the system can be computed as:

$$p^{(t+1)}_{FSM_j} = \sum_{FSM_i} p^{(t)}_{FSM_i} \frac{F(FSM_i)}{\overline{F}(t)} p_{FSM_i,FSM_j} \text{ (after the analysis in [Mic96])}$$

where proportions of $p^{(t)}_{FSM_j}$ of all FSMs FSM_i at time t are considered. These proportions are modified by fitness selection, where $F(FSM_i)$ is the fitness of FSM_i and $\overline{F}(t)$ is the average fitness of the population at time t. p_{FSM_i,FSM_j} computes the probability of mutating FSM_i to FSM_j. The result is the expected proportion of FSM_j at time t+1; mutation is carried out by one of the mutation operator discussed.

The total system is described by C^X equations, one for each FSM_j. Starting with initial population proportions $p^{(0)}_{FSM_j}$, the C^X equations are iterated repeatedly to produce the expected time evolution of the system. Fitness functions F can be aggregated to simplify the model. This is accomplished by reducing the number of equations and number of terms in an equation. Let the alphabet (processes or tasks) be denoted by Γ and let $\Lambda \in \Gamma$ be one of the C alleles. Let Λ^c denote all other alleles. The set of FSMs with j $\Lambda's$ form an equivalence class, and it suffices to have only X+1 equations, since there can be any where from zero to X $\Lambda's$ in an average FSM. This is a dramatic reduction from the C^X equations that would be required in the general case. Thus, the previous equation can be interpreted as FSM_i referring to any FSM with i $\Lambda's$ and FSM_j as any FSM with j $\Lambda's$. The probability of mutating any FSM with i $\Lambda's$ to one with j $\Lambda's$ is given by p_{FSM_i,FSM_j}. Suppose $j \geq i$ meaning that we are increasing or not changing the

number of a process Λ. Thus, mutation requires j-i more $\Lambda^{c\prime}s$ to be mutated to $\Lambda's$ than $\Lambda's$ mutated to $\Lambda^{c\prime}s$. The mutation probabilities are:

$$P(p_{FSM_i,FSM_j}|X=x)=\sum_{k=0}^{min\{i,x-j\}}\binom{i}{k}\binom{x-i}{k+j-i}p_m^k(1-p_m)^{j-k}\left(\frac{p_m}{C-1}\right)^{k+j-i}\left(1-\frac{p_m}{C-1}\right)^{x-j-k}$$

after the analysis in [Spe98], where k is the number of $\Lambda's$ mutated to $\Lambda^{c\prime}s$. Since there are i $\Lambda's$ in the current FSM, this means that i-x $\Lambda's$ are not mutated to $\Lambda^{c\prime}s$. This occurs with probability $p_m^k(1-p_m)^{j-k}$. Since k $\Lambda's$ are mutated to $\Lambda^{c\prime}s$ then k+j-i $\Lambda^{c\prime}s$ must be mutated to $\Lambda's$. Also, since there are x-i $\Lambda^{c\prime}s$ in current FSM, this means that x-i-k-j+i=x-k-j $\Lambda^{c\prime}s$ are not mutated to $\Lambda's$. This occurs with probability $\left(\frac{p_m}{C-1}\right)^{k+j-i}\left(1-\frac{p_m}{C-1}\right)^{x-j-k}$.

Thus, this yields the number of ways to choose k $\Lambda's$ out of i $\Lambda's$ and the number of ways to choose k+j-i $\Lambda's$ out of the x-i $\Lambda^{c\prime}s$. It is therefore clear that it isn't possible to mutate more than x-i $\Lambda^{c\prime}s$, $k+j-i\le x-i$ indicating that $k\le x-j$. The minimum of i and x-j bounds the summation correctly. If X is considered as a random variable with probability mass function (p.m.f) $p(x)$, then $P=\sum_{\forall x}P(p_{FSM_i,FSM_j}|X=x)p(x)$

Conversely, if $i\ge j$, we are decreasing or not changing the number of $\Lambda's$. The mutation probabilities are:

$$P(p_{FSM_i,FSM_j}|X=x)=\sum_{k=0}^{min\{x-i,j\}}\binom{i}{k+i-j}\binom{x-i}{k}p_m^{k+i-j}(1-p_m)^{j-k}\left(\frac{p_m}{C-1}\right)^{k}\left(1-\frac{p_m}{C-1}\right)^{x-i-k}$$

When $0.0<p_m<1.0$, all p_{FSM_i,FSM_j} entries are non-zero and the markov chain is ergodic. Thus there is a steady state distribution describing the probability of being in each state after a long period of time. By the definition of steady-state distribution, it can not depend on the initial state of the system. Thus, the initial state population has no effect on the long term behaviour of the system. The steady-state distribution reached by this markov chain model can be thought of as a sequence of X Bernoulli trials with success probability $1/C$. (after the analysis in [Spe98]) The steady state distribution can therefore be described by the binomial distribution giving the probability of being in state i (i.e. the probability that i $\Lambda's$ appear at a locus after a long period of time):

$$\lim_{t\to\infty}P(FSM_t=FSM_i)=\binom{X}{i}\left(\frac{1}{C}\right)^i\left(1-\frac{1}{C}\right)^{X-i}$$

On average, $\lim_{t\to\infty} P(FSM_t = FSM_i) = \frac{X}{C}$ with a variance $X\left(\frac{1}{C}\right)\left(1-\frac{1}{C}\right)$. If a 95% confidence interval is considered, then the limit is taken with $\pm 1.96\sqrt{\frac{X}{C}\left(1-\frac{1}{C}\right)}$. If X is large enough, and $1/C$ is small enough, then the limit is approximated as a poisson distribution equal to $\left(X\ \frac{1}{C}\right)$. That's to say, it has a mean value X/C. It is thus noted that the steady state distribution is dependent on the cardinality C rather than on mutation rate p_m or initial population. However, p_m and initial population do have an effect on the *rate* this limiting distribution is approached. With arbitrary C, this would be analogous to having a large number of $\Lambda's$ mutate to $\Lambda^c's$. Let $p_\Lambda^{(t)}$ be the expected proportion of $\Lambda's$ at time t. Then the expected time evolution of the process can be described by the differential equation:

$$\frac{d\ p_\Lambda^{(t)}}{dt} = -\ p_m\ p_\Lambda^{(t)} + \left(\frac{p_m}{C-1}\right)\left(1-\ p_\Lambda^{(t)}\right) = \left(\frac{p_m}{C-1}\right)\left(1-C\ p_\Lambda^{(t)}\right)$$

where $p_m\ p_\Lambda^{(t)}$ represents the *loss* which occurs if Λ is mutated; while the other term is the *gain* which occurs if an Λ^c is mutated to Λ. At steady state, the differential equation must be equal to zero and is satisfied by $p_\Lambda^{(t)} = 1/C$ which is expected. The general solution to the differential equation is
:

$$p_\Lambda^{(t)} = \frac{1}{C} + \left(p_\Lambda^{(0)} - \frac{1}{C}\right)e^{-\left(C\ p_m t\right)/(C-1)}$$

Thus, the population converges to a limiting state distribution which is quite likely to be the optimum solution. As $t \to \infty$, then $p_\Lambda^{(t)} = \frac{1}{C}$ which is an expected result. Other similar analyses can be found in [Bar97] [Cru99] [Fog95] [Spe98].

7 Conclusion

The analysis forwarded concludes with the fact that a limiting state is finally reached. This state is independent of the initial population and the mutation rate, but dependent on the cardinality of the alphabet of the driving L-system. The analysis was carried out using tools from compilers theory, game theory and markov chains. The limiting state was proven to be the optimal solution for a number of iterations reaching infinity.

References

[Aho86] Aho, A. V.; Sethi, R.; Ullman, J. (1986). Compilers: Principles, Techniques, and Tools. Addison-Wesley.

[Bad98] Badr, Amr (1998). "A Hybrid Framework for Optimal System Design", PhD diss., Cairo Univ.

[Bad99] Badr, A.; Farag, I.; Eid, S. (1999). "A Hybrid Evolutionary Approach to Intelligent System Design", Mathware & Soft Computing, v6, n1.

[Bak87] Baker, J. (1987). "Adaptive Selection Methods for Genetic Algorithms", in Proc. of the 2nd International Conference on Genetic Algorithms.

[Bar97] Barnett, L. (1997). "Tangled Webs: Evolutionary Dynamics on Fitness Landscapes with Neutrality", MSc diss., Univ. of East Sussex.

[Bar91] Bartak, J.; Herrmann, L.; Lovicar, V.; Vejvoda, O. (1991). Partial Differential Equations of Evolution, Ellis Horwood.

[Bat93] Battiston, E.; De Cindio, F. (1993). "Class Orientation and Inheritance in Modular Algebraic Nets", International Conference on Systems, Man, and Cybernetics, Conference Proceedings v2, pp717-723.

[Cru99] Crutchfield, J.; Newegen, E. (1999). "The Evolutionary Unfolding of Complexity", in Landweber; Winfree, E.; Lipton, R.; Freeland, S. (eds.) (1999). Evolution as Computation.

[Fog95] Fogel, D. (1995). Evolutionary Computation, IEEE Press.

[Gio94] Giordana, A.; Neri, F.; Saitta, L. (1994). "Formal Models of Selection in Genetic Algorithms", in Proc. of ISMIS 1994 vol. LNAI-869.

[Hof88] Hofbauer, J.; Sigmund, K. (1988). The Theory of Evolution and Dynamical Systems: Mathematical Aspects of Selection, Cambridge University Press.

[Ioo90] Iooss, G.; Joseph, D. (1990). Elementary Stability and Bifurcation Theory, 2nd Edn, Springer Verlag.

[Lou97] Louden, K. C. (1997). Compiler Construction: Principles and Practice. PWS Publishing Company.z

[Mic96] Michalewicz, Z. (1996). Genetic Algorithms + Data Structures = Evolution Programs, 3rd Edn., Springer Verlag.

[Spe98] Spears, W. (1998). " The Role of Mutation and Recombination in Evolutionary Algorithms", PhD diss., George Mason Univ.