

**ANALYZING PROGRAM DYNAMICS  
THROUGH  
STOCHASTIC L-SYSTEMS**

*A thesis submitted*

*for the degree of*

**Doctor of Philosophy**

*in*

**Computer Science**

*by*

**C. RAMESH KUMAR REDDY**



Department of Computer and Information Sciences  
School of Mathematics and Computer/Information Sciences

University of Hyderabad

Hyderabad - 500 046.

INDIA

2005



**Department of Computer and Information Sciences**  
**School of Mathematics and Computer/Information Sciences**  
**University of Hyderabad, Hyderabad, INDIA.**

---

### **CERTIFICATE**

This is to certify that the thesis work entitled '**Analyzing Program Dynamics through Stochastic L-Systems**' *being* submitted by **Mr. C. Ramesh Kumar Reddy** (Reg. No. **99MCPC01**) in fulfillment of the requirement for the award of degree of **Doctor of Philosophy (Computer Science)** of the University of Hyderabad, is a record of bona fide work carried out by him under my supervision.

The matter embodied in this thesis has not been submitted for the award of any other research degree.

**Dr. Chakravarthy Bhagvati**  
**(Supervisor)**

Department of Computer and Information Sciences,  
University of Hyderabad, Hyderabad, INDIA.

**Prof. Arun Agarwal**  
**(Head)**

Department of Computer and  
Information Sciences,  
University of Hyderabad,  
Hyderabad, INDIA.

**Prof. Arun K. Pujari**  
**(Dean)**

School of Mathematics and  
Computer/Information Sciences,  
University of Hyderabad,  
Hyderabad, INDIA.

## **DECLARATION**

I, **C. Ramesh Kumar Reddy**, hereby declare that the work presented in this thesis has been carried out by me under the supervision of **Dr. Chakravarthy Bhagyati**, Department of Computer and Information Sciences, University of Hyderabad, Hyderabad, India, as per the PhD ordinances of the University. I declare, to the best of my knowledge, that no part of this thesis has been submitted for the award of a research degree of any other University.

**C. Ramesh Kumar Reddy**  
**(99MCPC01)**

# **DEDICATION**

If you want to dedicate your work, then only this page is necessary.

## ACKNOWLEDGEMENTS

‘Guru’ – the person who removes the ignorance and shows the path to knowledge. Once I look back over the years, my first contact with my Guru Dr. Chakravarthy Bhagvati to till date are live and vivid in my memory.

My special acknowledgements go to Dr. Arun K Pujari, Dr. Arun Agrawal and Dr. Hrishikesh Mohanthy for their invaluable support and suggestions. Also I acknowledge the contributions from Dr. Bapiraju and Dr. Durga Bhavani.

I am proud to have students like Miss K. Komal, Meera, KK, Raj, Sita, Sujit, Srinivas Kalaval and TNC Karthik, without their efforts this work would not have progressed this far.

There is a woman behind every successful man. I am fortunate to have Sobha as my better half (of life). In one word she is ‘Karyeshu Manthri’. Also I am indebted to my parents Smt. C. Anasuya and Sri C. Chandra Sekhar Reddy, without their support I would not have been progressed this far in my life.

# TABLE OF CONTENTS

<b>Certificate .....</b>	<b>i</b>
<b>Declarationdd .....</b>	<b>ii</b>
<b>Dedication .....</b>	<b>iii</b>
<b>Acknowledgements .....</b>	<b>iv</b>
<b>List of Figures .....</b>	<b>viii</b>
<b>List of Tables .....</b>	<b>x</b>
<b>Abstract.....</b>	<b>xi</b>
<b>1      <b>Introduction .....</b></b>	<b>1</b>
1.1 Scope.....	2
1.2 Motivation .....	3
1.3 Contributions .....	3
1.4 Organization of the thesis .....	3
1.5 Program dynamics through stochastic L-systems .....	3
1.6 Generating L-systems to model the program dynamics .....	5
1.7 Growth of Strings .....	6
<b>2      <b><i>Program Dynamics</i> .....</b></b>	<b>7</b>
2.1 Static Analysis .....	7
2.1.1 Static Analysis techniques .....	7
2.1.2 Advantages and Limitations of Static checking.....	10
2.2 Dynamic Analysis.....	11
2.2.1 Tools for Dynamic Analysis .....	11
2.2.2 Advantages and Limitations of Dynamic checking .....	14
2.3 Visualization.....	15

2.3.1	Visualization Tools.....	17
2.3.2	Advantages and Limitations of Visualization.....	17
<b>3</b>	<b>Program dynamics through Stochastic L – systems.....</b>	<b>19</b>
3.1	L-systems .....	19
3.2	Stochastic L-Systems.....	20
3.3	Visualizing dynamic behaviour .....	21
3.3.1	Mapping of Block-structured language to L-Systems.....	22
3.3.2	Input analysis .....	25
3.3.3	Turtle graphics .....	26
3.4	Implementation.....	27
3.4.1	Program visualization.....	27
3.5	Case study : Visualization of Quick Sort .....	31
3.5.1	Graphical analysis .....	41
3.6	Summary.....	42
<b>4</b>	<b>Generating L-Systems to model the Program Dynamics .....</b>	<b>44</b>
4.1	Introduction.....	44
4.2	Methodology .....	44
4.2.1	Algorithm .....	45
4.3	Case study.....	47
4.3.1	Result – 1(Fibonacci Series).....	47
4.3.2	Result – 2(Growth of plant algae).....	49
4.4.3	Result – 3(Thue Morse code).....	49
4.4	Limitations.....	50
4.5	Visualization	
.....		50
4.6	Discussions .....	51
4.7	Conclusions.....	52

<b>5</b>	<b>Properties of strings.....</b>	<b>53</b>
5.1	Strings in DOL systems .....	53
5.1.1	Growth Functions.....	54
5.1.2	Limitations.....	54
5.2	Growth functions in stochastic L-systems .....	55
5.2.1	Multinomial Distribution.....	55
5.2.2	Length of the strings generated.....	55
5.2.3	Strings distribution at a given level .....	58
5.2.4	Symmetric properties .....	59
5.3	Results and discussions.....	62
5.4.	Limitations.....	72
5.5.	Conclusions .....	72
<b>6</b>	<b>Conclusions .....</b>	<b>73</b>
	<b>References.....</b>	<b>75</b>

## LIST OF FIGURES

1.1(a) Tree generated for Quick Sort (Descending)	4
1.1(b) Visualization of Quick Sort (Random)	4
1.1(c) Visualization of Quick Sort (Ascending)	4
1.2 Block diagram of reverse approach	5
2.1 An example of Floyd's interpreted program	8
3.1 Example of a derivation in L-system	19
3.2 Stochastic branching structure	21
3.3 Static branching structure of a program	23
3.4 Mapping of block structured language to L-systems	24
3.5 Mapping of a sample 'C' program to L-systems	25
3.6 Turtle graphics string interpretation commands	26
3.7 Tree generated for Student grade program( Sample data : 1)	29
3.8 Tree generated for Student grade program( Sample data : 2)	30
3.9 Mapping of Quick Sort to L-systems.	33
3.10a Tree generated for random data-Quick Sort (Normal C program) (Input: 32, 12, -9, 5, 13, 18)	36
3.10b Tree generated for random data-Quick Sort (stochastic L-system program) (Input: 32, 12, -9, 5, 13, 18)	37
3.11a Tree generated for random data-Quick Sort (34,23,41,2,3,25,20)	37
3.11b Tree generated by stochastic L-system for random data 34,23,41,2,3,25,20	38
3.12a Tree generated for ascending data-Quick Sort	39
3.12b Tree generated by stochastic L-system for ascending data.	39
3.13a Tree generated for descending data-Quick Sort 7, 6, 4, 3, 2, 1	40
3.13b Tree generated by stochastic L-system for descending data 7, 6, 4, 3, 2, 1	40
3.14 Graph showing the no. of occurrences of major steps in quick sort against various probability distributions	41
4.1 Over view of algorithm reverse process	44
4.2 Growth of plant algae <i>Chaetomorpha linum</i>	50

5.1	Trees obtained for the stochastic L-system $D \rightarrow D + D/D - D/DD$ with probabilities $2/12$ , $4/12$ and $6/12$ respectively	64
5.2	Trees obtained for the stochastic L-system $D \rightarrow D + D/D - D/DD$ with probabilities $4/12$ , $4/12$ and $4/12$ respectively.	65
5.3	Trees obtained for the stochastic L-system $D \rightarrow D + D/D - D/DD$ with probabilities $9/12$ , $2/12$ and $1/12$ respectively	66
5.4	Trees obtained for the stochastic L-system $D \rightarrow D + [D + D]/D - D/DD$ with probabilities $2/12$ , $4/12$ and $6/12$ respectively.	69
5.5	Trees obtained for the stochastic L-system $D \rightarrow D + [D + D]/D - D/DD$ with probabilities $4/12$ , $4/12$ and $4/12$ respectively	70
5.6	Trees obtained for the stochastic L-system $D \rightarrow D + [D + D]/D - D/DD$ with probabilities $9/12$ , $2/12$ and $1/12$ respectively	71

## LIST OF TABLES

3.1	Table illustrating the program dynamics of Quick Sort	35
4.1	Trace of Reverse Process Algorithm	48
5.1	Table showing the length and strength of stochastic L-system Rule set1 with probability $2/12$ , $4/12$ and $6/12$	62
5.2	Theoretical and empirical results of Rule set $D \rightarrow D + D / D - D / DD$ with probability of $2/12$ , $4/12$ and $6/12$	63
5.3	Table showing the length and strength of stochastic L-system Rule set1 with equal probability	65
5.4	Table showing the length and strength of stochastic L-system Rule set1 with probability of $9/12$ , $2/12$ and $1/12$	66
5.5	Table showing the length and strength of stochastic L-system Rule set2 with probability of $2/12$ , $4/12$ and $6/12$ .	67
5.6	Table showing the length and strength of stochastic L-system Rule set2 with probability of $4/12$ , $4/12$ and $4/12$ .	67
5.7	Table showing the length and strength of stochastic L-system Rule set2 with probability of $9/12$ , $2/12$ and $1/12$ .	68
5.8	Table showing the length and strength of stochastic L-system Rule set3 with probability of $2/12$ , $4/12$ , $6/12$ .	68
5.9	Table showing the length and strength of stochastic L-system Rule set3 with probability of $4/12$ , $4/12$ , $4/12$ .	70
5.10	Table showing the length and strength of stochastic L-system Rule set3 with probability of $9/12$ , $2/12$ , $1/12$ .	71

## ABSTRACT

One of the major areas of research in software engineering is *program dynamics* i.e., the run-time behavior of a computer program. In this thesis, we propose a novel scheme for analyzing run-time behavior of programs. Our proposed method is strongly linked to visualization.

The dynamic evolution of a program resembles a tree with the main function as the root and other block / function invocations as branches. L-systems are mathematical models to generate trees and in our thesis we develop an approach that models a program as a stochastic L-system. The probabilities associated with generators are derived from knowledge of input distributions.

We illustrate our approach on the well-known quick-sort algorithm. Its time-complexity and other features are related to both qualitative ( such as branching factor, symmetry and size ) and quantitative measures ( such as probabilities and lengths ) on the tree-like visualizations. Specifically, we demonstrate that our tree visualization is capable of capturing the different behaviours of quick-sort on randomly-ordered and already sorted input lists.

One of the major contributions of our work is the quantitative analysis possible in our visualization scheme. We derive quantitative solutions to the following three questions:

- 1) Given an L-system, what are the minimum and maximum sizes of the trees generated for specific input sizes?
- 2) Given an L-system and a tree obtained by visualizing the program behaviour for a specific input, what is the probability of obtaining such a tree?
- 3) Given a specific L-system and an input distribution which functions/ blocks are likely to be called more often?

We also propose and demonstrate a preliminary approach to derive L-system rules from observed outputs of a program. Such an approach is useful in analyzing the dynamic behaviour in the absence of source code or static structure of the program.

## CHAPTER 1

### INTRODUCTION

One of the major areas of research in software engineering is *program dynamics* i.e., the study of run-time behaviour of a computer program. Program dynamics indicates the flow of control in a program, what values are computed, the memory usage, the execution time of a program and others [44, 81].

A typical program consists of several branch and loop control structures resulting in hundreds of paths between input and output. For example, even an extremely simple program that converts a letter grade (such as 'A, B, C, D' or 'F') into grade point average has a switch statement inside a double loop structure. The switch statement has as many branches as there are grades, and the loops iterate over the number of students and the number of subjects. The actual paths taken by inputs depend on the specific grades awarded and their distribution.

A complete analysis of the run-time behaviour is possible only in simple programs. As the dynamics depend on inputs, any analysis requires knowledge of inputs. Even for a simple program, it is not possible to consider all the input data combinations. A way out is to assume different distributions such as Normal, Poisson, Uniform etc., for inputs and perform a statistical analysis.

Any program starts with a main function and takes different branches depending on the input values. Each branch leads to further computations, modifications of intermediate data and further branching. The end result is that the program evolution resembles an inverted tree that starts at a *root* (the main function) and then grows into a large, complex tree. Loop statements are unraveled as the program evolves and contribute to larger sub-patterns within the complete tree structure. Note that *tree* refers to the arboreal variety and not the tree data structure.

L-Systems, proposed by Lindenmayer [85], were originally designed to replicate the wide variety of trees, plants and shrubs seen in nature. L-systems comprise a set of rewriting rules that are interpreted as instructions in turtle graphics [11,12,95,113]. Various types of L-systems are available. Of particular interest are stochastic L-systems

that allow introduction of variations in the rigid tree structures ‘grown’ by pure L-systems [34,137].

In this thesis, a novel scheme is proposed for analyzing the program dynamics that is strongly linked to visualization. Here stochastic L-systems are mapped to program dynamics for both visualization and analysis. The visualized tree behaviour is qualitatively analyzed. Qualitative analysis of the tree-like run-time structure reveals useful information about the program. A tree of narrow branching indicates that the data path traversal is through only a portion of the program. A tree of wide branching means all the blocks of the program have been traversed uniformly. Similarly, a left or right heavy tree indicates that only subsets of functions are executed.

The qualitative features of the tree like structure of the visualized program can be analyzed mathematically in our approach because of their basis in L-systems. L-systems, offers several advantages over other visualization tools such as like *Hot Wired* [76], *JA Vis* [97], *Rigi* [102] and *ALMA* [114] etc. The central idea in this thesis is to model the programs’ tree growth mathematically so that it is amenable to *qualitative* and *quantitative analysis*. The aim is to analyze qualitatively and quantify the observed features of program dynamics through the use of visualization techniques.

## 1.1 Scope

The use of mathematically rigorous and well-researched L-systems in visualizing program dynamics permits us to measure certain aspects of program’s behaviour. For example, the general shape of the tree, whether it is left or right heavy or symmetric indicates which subset of functions or blocks is exercised.

The length of a branch indicates the complexity or repetition of a particular block where as the height indicates the number of invocations of various blocks/ functions. The tree complexity may thus be estimated from the size of the tree.

Our research develops a methodology for mapping L-systems to block-structured programs and their subsequent visualization as trees. We describe various qualitative features and derive quantitative measures using L-systems theory for assistance in developing test sets and computing time complexities. We also suggest that the tree like visualizations may help reorganize programs for better performance.

## **1.2 Motivation**

Until now, little quantitative analysis of program dynamics was done using visualization techniques. In this regard we made a successful initial attempt in visualizing and quantifying the program dynamics using stochastic L-systems.

## **1.3 Contributions**

Our contributions are

- Visualizing program dynamics using L-systems.
- Qualitative and quantitative analysis of program dynamics.
- L-systems based parameters for software testing.

## **1.4 Organization of the thesis**

The rest of this Chapter is an overview of the thesis. In section 1.5, we present the highlights of the approach described in Chapter 3. Section 1.6 is about obtaining the original rule set given two successive stages of program evolution. This reverse approach is explained in detail with the help of an algorithm in Chapter 4. Section 1.7, deals with the properties of the trees generated by our approach. The strings generated by L-systems are visualized as trees. Throughout this thesis, we treat strings and trees synonymously. The strings are qualitatively analyzed for their properties such as length, distributions and symmetry etc. These and other string properties are discussed in Chapter 5.

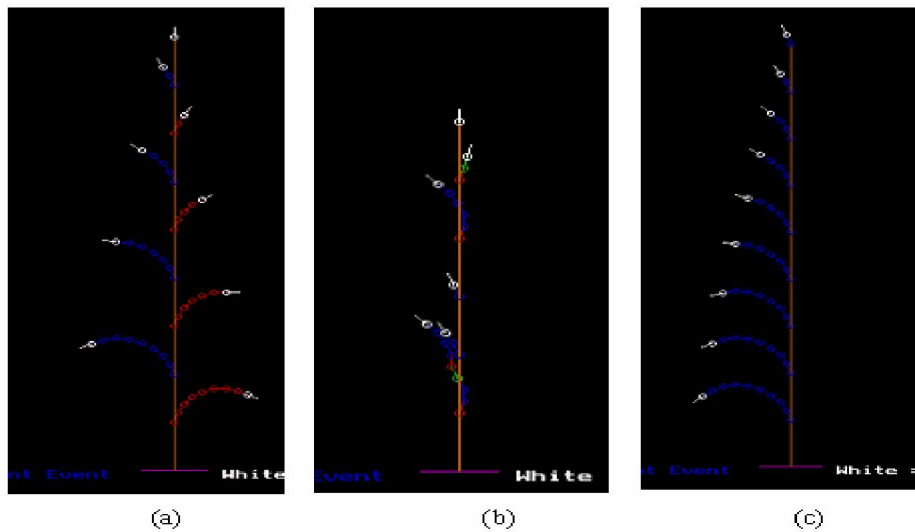
## **1.5 Program dynamics through stochastic L-systems**

The runtime behaviour of a program resembles that of an evolution of a tree. L-systems and stochastic L-systems are well studied by scientists and mathematicians alike. A significant amount of research has gone into the field of mathematical properties of L-systems and quantified by scientists like Doucet P., Rozenberg G., Saloma A and Vitanyi etc. [135–137,139,140,163,164]. This has prompted us to choose stochastic L-systems for modeling tree-like representations of program dynamics.

In our method of program visualization there is a mapping of given program to stochastic L-systems. The basic block of a program is visualized using L-systems turtle graphics.

Analysis of quick-sort is used as a case study to illustrate our approach. There are three main operations in each recursive invocation of quick-sort algorithm function. We visualize these three operations using L-system rules and turtle graphics. The first operation, which increments the lower bound of the list, is drawn as a branch to the right. The second, which decrements the upper-bound, is drawn as a branch to the left. The third, which swaps the pivot element, is a straight-line.

The visualization of quick-sort program yields tree like structures as shown in Figure 1.1.



**Figure 1.1:** Various trees generated for quick-sort program.

Figure 1.1 shows the resulting trees from different runs of quick-sort program on three sets of inputs. Figure 1.1(a) is the result from running the program on a list that is in descending order. It is obvious that the lower and upper bounds are updated alternately. In Figure 1.1(b), the input is randomly generated and it may be seen that there is no clear pattern in the branching. In Figure 1.1(c), the tree contains only left branching (i.e., only the upper bound is changed) and is the result of an input that is already sorted. Further, it may be noticed that both the height and width of the trees Figure 1.1 (a) and Figure 1.1(c)

are larger than that of Figure 1.1(b), and is indication of greater time-complexity,  $O(n^2)$ , in such cases as opposed to the average time complexity of  $O(n \log n)$  as is the case of Figure 1.1(b). Detailed analysis is described in Chapter 3.

## 1.6 Generating L-systems to model the program dynamics

In the above analysis of quick-sort program, the L-system was derived from the program structure given to us. When such information is lacking, the program may be run repeatedly for any given input. The output observations are captured. Then we devise an algorithm to estimate the equivalent L-system rules that could have resulted in the observed sequence of outputs.

In our approach given two successive stages of a program output, an attempt has been made to get back the L-system rule-set which is responsible for program visualization. This is illustrated in Figure 1.2. For this purpose we have taken two successive strings  $Str_i$  and  $Str_{i+1}$  generated in forward approach.



**Figure 1.2:** Block diagram of reverse approach

Let us consider the two strings ‘*abbab*’ and ‘*bababbab*’. These two strings were given as input to the ‘*Reverse Approach*’ process. After successful processing we get the *rule set* responsible in generating the above strings. The rule set thus obtained is given below. The process of obtaining the rule set, given successive strings is discussed in detail in Chapter 4.



This approach of finding the ‘rules’ responsible for generation of strings(objects), helps us in better understanding of the source code.

## 1.7 String Properties

The strings generated in the forward approach using stochastic L-systems were analyzed. Given an L-system and a particular level of string (tree) generation, properties such as string length at a specified level, string distribution at that level, probability of arriving at a string and any symmetric property of the generated trees are computed.

The quantitative analysis of strings is carried out with the help of a mathematical model, *Multinomial Distribution*, which is a generalization of Binomial distribution [149,151]. Unlike Binomial theorem, which deals with two variables, Multinomial theorem handles more than two variables.

Using the above model, an attempt has been made to analyze the tree behaviour. The string properties are quantified in Chapter 5.

In the next chapter, a detailed description of evolution of program dynamics and various types of visualization techniques that are significant to our approach are discussed.

## REFERENCES

- [1] Ammons G. and Larus J.R., (1998), "Improving Data-flow Analysis with Path Profiles", *Proceedings of the SIGPLAN '98 Conference on Programming Language Design and Implementation, Montreal, Canada*, pp. 72–84.
- [2] Ammons G., Ball T., and Larus J.R., (1997), "Exploiting Hardware Performance Counters with Flow and Context Sensitive Profiling", *Proceedings of the SIGPLAN '97 Conference on Programming Language Design and Implementation, Los Vegas*, pp.85-96.
- [3] Autrey C. Pu, Black A., Consel C., Cowan C., Inouye J., Kethana L., Walpole J. and Zhang K., (1995), "Optimistic Incremental Specialization: Streamlining a Commercial Operating System", *Proceedings of the Fifteenth ACM Symposium on Operating System Principles, Copper Mountain Resort, CO*, pp. 314-324.
- [4] Backus J. W., (1959), "The syntax and semantics of the proposed international algebraic language of the Zurich ACM-GAMM conference", *Proc. Intl. Conf. on Information Processing, UNESCO*, pp. 125-132.
- [5] Baker B.S., (1995), "Parameterized Duplication in Strings: Algorithms and an Application to Software Maintenance", *SIAM Journal of*, (26), pp. 1343-1362.
- [6] Bala V., (1996), "*Low Overhead Path Profiling*", Technical report, Hewlett Packard Labs.
- [7] Ball Thomas, (1994), "Efficiently counting program events with support for on-line queries", *ACM Transactions on Programming Languages and Systems*. 16(5), pp.1399-1410.
- [8] Ball Thomas and James R. Larus, (1994), "Optimally profiling and tracing programs", *ACM Transactions on Programming Languages ad Systems*, 16(4), pp.1319-1360.
- [9] Ball. T. and Larus J.R., (1996), "Efficient Path Profiling", *Proceedings of the 29<sup>th</sup> Annual IEEE/ACM International Symposium on Microarchitecture, Paris, France*, pp. 46-57.
- [10] Ball T., (1999), "The concept of dynamic analysis", *Proceeding of 7<sup>th</sup> European Software Engineering Conference*, pp. 216-234.