Research Paper Presentation

Avula Mohana Durga Dinesh Reddy

CS20BTECH11005

Title and authors

Title

PROBABILITY REGRESSION SUITES

Authors

1. Shai Fine,fshai@il.ibm.com, IBM Research Laboratory in Haifa ,31905 ,Israel
2. Shmuel Ur, ur@il.ibm.com ,IBM Research Laboratory in Haifa ,31905 ,Israel
3. Avi Ziv ,aziv@il.ibm.com, IBM Research Laboratory in Haifa ,31905 ,Israel

Abstract

1. Random test generators are often used to generate regression suites for testing.
2. Regression suites are commonly generated by choosing several specifications and generating a number of tests from each one, without reasoning which specification should be used and how many tests to be generated from each specification.
3. This paper describes about a practical technique for building high quality regression suites, this technique is shown to be cheaper and efficient from experiments.

Concepts and definitions

- **Regression suite** : It is a set of test specifications.
- **Optimisation Problem** : It is the set of dersired conditions the Probabilistic Regression suite is requred to satisfy.

Abbreviations

Abbreviation	Formula	Meaning
t	$=(t_1, ...t_n)$	the set of tasks to be covered
s	$=(s_1, ..., s_k)$	the sets of parameters that are allowed to be used in stimulation runs.
P_j^i		The probability of covering task t_j in a stimulation that uses parameter set(*testspecification*) s_j
w_i	$w_i \in \mathbb{N}$	which denotes the number of times a stimulation, using parameter set s_i must be activated.
policy w	$=(w_1, ..., w_k)$	the activation policy
W	$\sum w_i$	denotes the total number of stimulation runs derieved by using policy w.
E_{cnst}		lower bound for expected coverage
ξ_i	> 0	additional set of non-negative slack variables for violation magnitude.
c_i		It is the cost of overall resource consumpltion ehile using the parameter s_i
C		The bound on the total resouce consumption

Table: Abbreviations used in presentation

Introduction

1. Presently, up to 70% of design development time and resources are spent on functional verification.
2. Current industrial practice uses stimulation-based verification (or dynamic verification) as the main functional verification vehicle for large and complex designs.
3. Regression testing plays an important role in dynamic verification.
4. In Regression testing , a set of tests known as *regression suite* ,is stimulated periodically and after major changes in the design or its environment , in order to check that no bugs were introduced.

Introduction Contd.

5. While using a random test generator, test suites don't have to be maintained ,only the generator is maintained for other reasons.As a result , random regression suites are often preferred over maintaining test suites.
6. An important source of information that can be used to create efficient random regression suites is the probability of each test specification to cover each of coverage tasks.Another source can be a Coverage Directed Generation Engine which provides estimates of these probabilities.
7. In this paper we will first show how to construct a regeression suite that uses minimal number of tests required to achieve a specific coverage goal.Then we show how to create a regression suite that maximizes coverage when fixed number of tests are used.

Constructing Probabilistic Regression Suite

1. Given a policy w , the probability of covering a task t_j is represented by

$$P_j = E(t_j) = 1 - \prod_i \left(1 - P_j^i\right)^{w_i} \tag{1}$$

and since the event of covering task t_j is Bernoulli , $P_j = E(t_j)$ is the expected coverage of task t_j.

2. Firstly lets define *optimisation problem* as :**Probabilistic Regression Suite**. *Find the policy w , which minimises the number of stimulation runs and with high probability provides a desired coverage.*

$$\begin{aligned} min_w \quad & \sum w_i \\ \text{such that } \forall j \quad & P_j = 1 - \prod_i \left(1 - P_j^i\right)^{w_i} \geq E_{cnst} \\ \forall i \quad & \mathbb{N} \ni w_i \geq 0 \end{aligned} \tag{2}$$

Constructing Probabilistic Regression Suite Contd.

2. This is a integer programming(IP) problem which is difficult to solve. So we use a relaxation technique by applying log.The resulting *optimisation problem* is linear.

$$\begin{aligned} min_w \quad & \sum w_i \\ \text{such that } \forall j \quad & \sum_i w_i \times g_{ij} \leq log\,(1 - E_{cnst}) \\ \forall i \quad & \mathbb{N} \ni w_i \geq 0 \end{aligned} \tag{3}$$

where $g_{ij} = log\,(1 - E_{cnst})$

3. Now lets focus on the tasks that have too small probabilities, the number of stimulation runs required to cover these tasks is too high so we will instead "charge" the objective function with violation magnitude times cost c , for each violation constraint.

Constructing Probabilistic Regression Suite Contd.

3. Lets add this property to the optimisation problem.
   The definition of *Optimisation problem* becomes :**Soft Probabilistic Regression Suite**. *Find the policy w , which minimises the number of stimulation runs and with high probability provides a coverage that permits violation of the lower bounds for task coverage.*

$$
\begin{aligned}
min_w \quad & \sum w_i + c \sum_i \xi_i \\
\text{such that } \forall j \quad & \sum_i w_i \times g_{ij} \leq log\,(1 - E_{cnst}) \\
\forall i \quad & \mathbb{N} \ni w_i \geq 0 \\
\forall j \quad & \xi_i \geq 0
\end{aligned} \tag{4}
$$

Constructing Probabilistic Regression Suite Contd.

1. Now lets add another property to optimisation problem which is slightly different yet very common scenario of *limited number of resources*.
   The definition of *Optimisation problem* becomes :**Expected Coverage Probability with**. *Given a bound on the number of stimulations permitted, W, and a bound on the cost of the resource consumption C, find the policy w, which maximises the expected coverage probability*

$$
\begin{aligned}
max_w \quad & \sum_j \left(1 - \prod_i \left(1 - P_j^i\right)^{w_i}\right) \\
\text{such that} \quad & \sum_i w_i \leq W \\
& \sum_i c_i w_i \leq C \\
\forall i \quad & w_i \geq 0
\end{aligned} \tag{5}
$$

Constructing Probabilistic Regression Suite Contd.

2. In the next step we focus on covering the least probable tasks to cover.So the definition of *Optimisation problem* becomes : **Least Probable Coverage Task with Limited Resources**. *Given a bound on the total number of stimulations W, and a bound on the cost of resource consumption C,find the policy w that maximizes the probability of the least probable task to cover*

$$\begin{aligned} max_w \quad & min_j \left(1 - \prod_i \left(1 - P_j^i\right)^{w_i} \right) \\ \text{such that} \quad & w_i \leq W \\ & \sum_i c_i w_i \leq C \\ \forall i \quad & w_i \geq 0 \end{aligned} \tag{6}$$

Experimiental Results

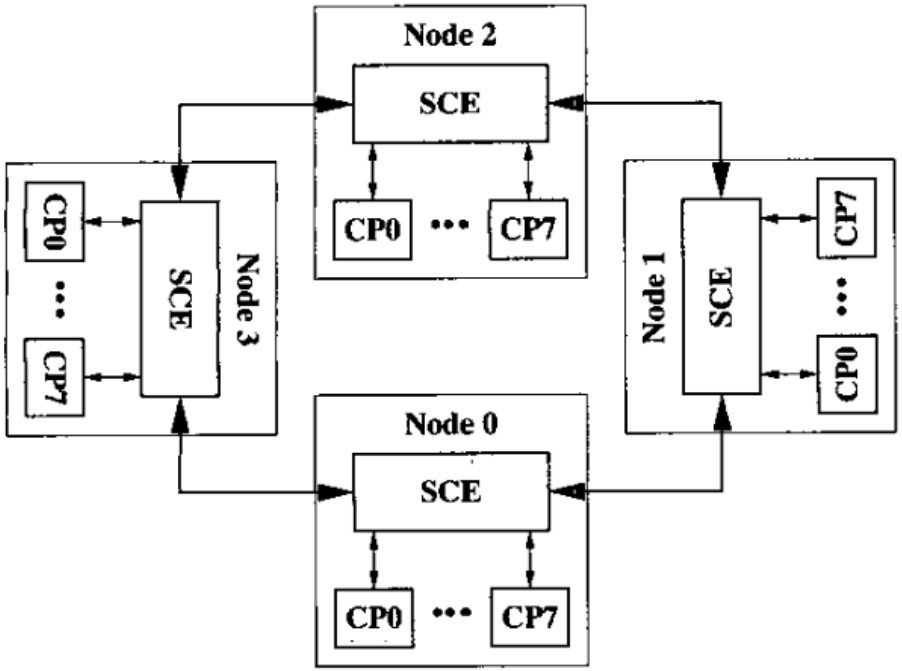

Figure: Structure of the SCE stimulation environment of an IBM z-series system

Regression suite for Hardware Verification with minimal Number of Stimulations

1. Many experiments were conducted of a coverage model used in the verification of the Storage Control Element (SCE) of an IBM z-series as shown in Figure 1.
2. It contains six attributes : the core that initiated the command, the pipeline in the SCE that handled it , the command itself, and three attributes that relate to the response.
3. In the first experiment ,we concentrated on a subset of the coverage model that deals with unrecoverable errors(UE).The size of UE is 98 all of ehich are hard to cover.

Regression suite for Hardware Verification with minimal Number of Stimulations Contd.

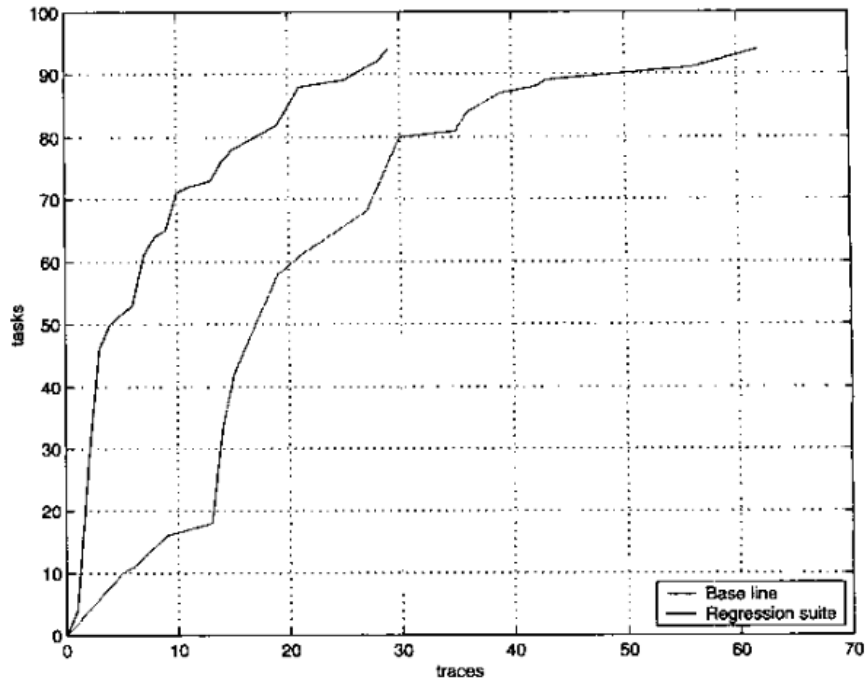

Figure: Coverage process of UE subset of the SCE

Regression suite for Hardware Verification with minimal Number of Stimulations Contd.

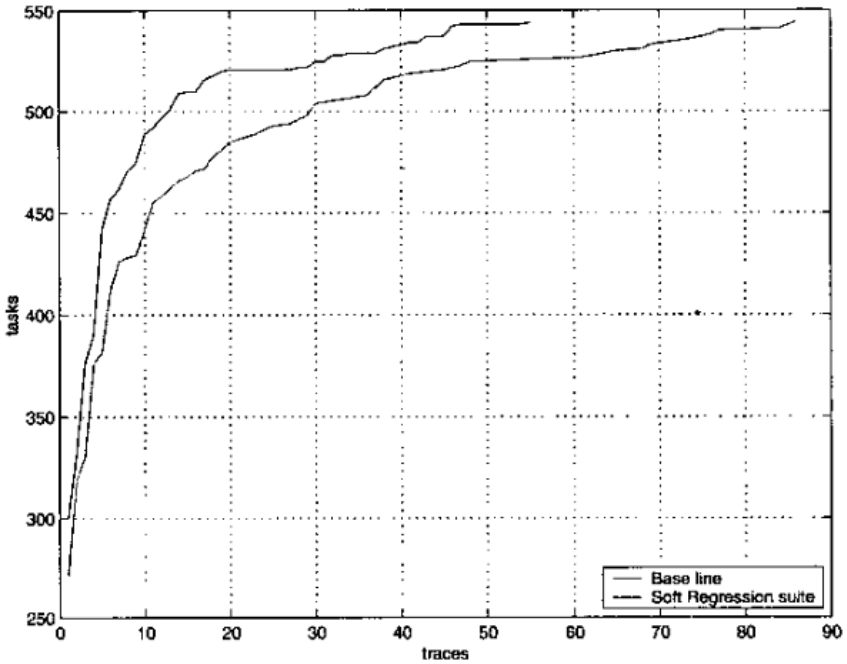

Figure: Coverage process of SCE model

Regression suite for Hardware Verification with minimal Number of Stimulations Contd.

4. Using definition Probabilistic Regression Suite with a lower bound of 1/2 for expected coverage it took only 29 stimulations to cover 94 events (96%). This when compared to Common practice toom 61 stimulations to cover the same coverage as shown in 2.
5. The second experiment tageted the whole SCE coverage model which has 564 tasks, most of which aren't hard to cover . The repository we used had 126 test specifications.
   In this experiment we used definition **Soft Probabilistic Regression suite** with a lower bound of 1/2 for expected coverage. It took only 55 stimulations to cover 554 events (98%). This when compared to activation of every parameter set one by one , it took 86 stimulations to reach same coverage as shown in 3.

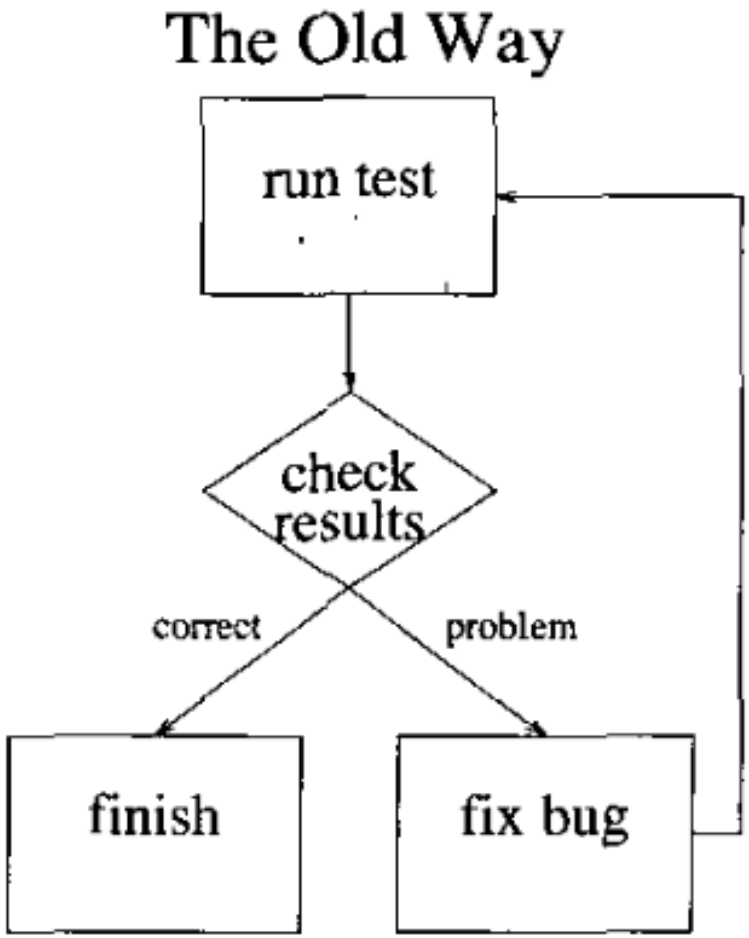

Figure: Each test executed once

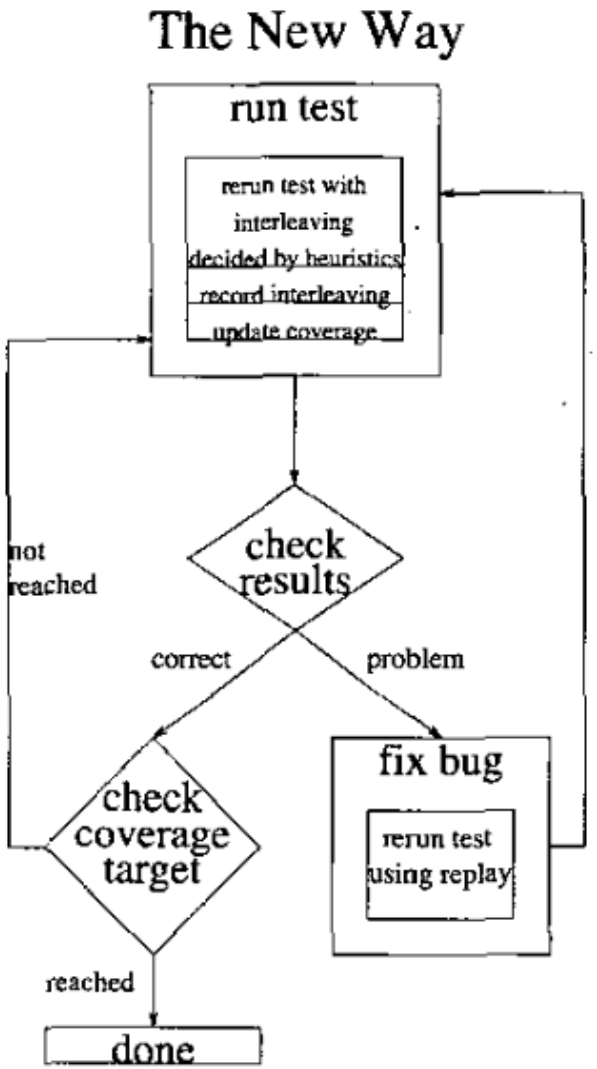

Figure: Executiong until reaching coverage target

Regression suite for Software Testing with a limited Number of Test Executions

1. A test in the multi-threaded domain is a combination of inputs and interleaving , where interleaving is the relative order in which the threads were executed. Running the same inputs twice may result in different outcomes.
2. The process depicted in 4 Load testing (i.e testing the applicatication under a real or stimulated workload), increases the likelihood of some interleaving that would be unlikely under light load. However , load testing is not systematic and is expensive and can be only executed at the end of the testing cycle.
3. ConTest is a tool for generating different interleaving for the purpose of revealing concurrent faults.ConTest takes a heuristic approach of seeding the program with instrumentation in concurrently significant locations.
4. The process of re-execcuting tests using ConTest is depicted in 5 . Each time the functional test t is run as a result od seding technique , ConTest produces a potentially different interleaving.

Regression suite for Software Testing with a limited Number of Test Executions Contd.

5. For the experiment we used 13 different heuristics as the test specifications.Given the statistics collected for the 13 heuristics we constructed policies designed to maximize the coverage of the 10,000 events using no more than 250 and 1000 test runs.We used definition **Expected Coverage with limited probability** while using greedy technique to get the solutions.
6. The policy was designed to yield best coverage with 250 runs using only 2 heuristics.The policy for the 1000 runs was constructed on 4 heuristcs, two of which are dominant(used roughly 83% of the time)

Regression suite for Software Testing with a limited Number of Test Executions Contd.

	Num. Events Covered	
	250	1000
Uniform Policy	1449	1988
Best Pure Heuristic	1699	2258
Greedy Policy	1723	2429

Table: Total coverage using ConTest for 250 and 1000 test runs

The first row shows the result averaged over all policies.Second row shows the result of the best policy.Third row shows the result generated by greedy policy.

7. table 2 shows the number of events covred using 250 ,1000 test runs. Note in passing every heuristic in the 1000 test runs and combinig the results (i.e a total of 13,000 test runs) , only 4338 events were covered .

Conclusions and Future work

1. We have sown that regression suites created are better than those generated at random.This is true for both limited resource and specific utilities scenarios.
2. Experiments shows that our technique is economically rewarding.
3. First we plan to investigate the use of the proposed technique to guide the entire functional verification process.
4. We also investigate the use of hybrid schemes that combines the building of probabilistic regression suites with snapshots of current coverage.
5. Special attention should be given to tasks that are coveraged with a very low probabilty.One possible solution to this issue is try not to cover these tasks using a random regression suite.Instead, keep tests that cover such tasks along with other tests that are kept in a deterministic regression suite.