Poster: Enhancing Partition Testing through Output Variation

Huai Liu
RMIT University
huai.liu@rmit.edu.au

Pak-Lok Poon
The Hong Kong Polytechnic University
pak.lok.poon@polyu.edu.hk

Tsong Yueh Chen
Swinburne University of Technology
tychen@swin.edu.au

Abstract—A major test case generation approach is to divide the input domain into disjoint partitions, from which test cases can be selected. However, we observe that in some traditional approaches to partition testing, the same partition may be associated with different output scenarios. Such an observation implies that the partitioning of the input domain may not be precise enough for effective software fault detection. To solve this problem, partition testing should be fine-tuned to additionally use the information of output scenarios in test case generation, such that these test cases are more fine-grained not only with respect to the input partitions but also from the perspective of output scenarios.

Index Terms—Partition testing, choice relation framework, output scenario.

I. INTRODUCTION

Partition testing is a popular approach to test case generation. It first divides the set of all possible program inputs (namely input domain) into disjoint partitions, and then selects at least one input from each partition to construct a set of test cases. Many software practitioners consider that a single input will be sufficient to represent a partition, if the partition is homogeneous [5]. Typical partition testing methods include the CHOिCe reLAिIon framEwork (CHOिC’LATE) [1][2], classification-tree method [4], and combinatorial testing [3].

However, we observe that, in some cases, the same input partition is associated with different output scenarios, indicating that some partitions are not sufficiently homogeneous. This problem jeopardizes the very benefit of using input partitions for generating test cases. To alleviate the above problem, we propose that the variations of output scenarios should also be considered when test cases are generated. In this paper, we make use of a typical partition testing method, namely CHOिC’LATE, to illustrate how the use of various output scenarios enhances partition testing.

II. CHOिC’LATE: A PARTITION TESTING METHOD

The purpose of CHOिC’LATE [1] [2] is to help testers generate test cases from specifications. It works as follows:
1. Identify categories and choices. Testers first identify input parameters and environment conditions as categories whose values or states affect the software execution behavior. Each category is further partitioned into choices, which refer to the category’s different cases.
2. Determine the relation between each pair of choices, and capture all these relations in a choice relation table.
3. Generate complete test frames from the choice relation table. Any combination of choices is referred to as a test frame. A test frame TF can be further defined to be complete, if a test case is generated by selecting a concrete value from each choice in TF.
4. For each complete test frame, construct a test case by selecting a concrete value for each choice in the frame.

III. MOTIVATING EXAMPLE

As mentioned above, many partition testing methods, including CHOिC’LATE, focus on how to partition the input domain so that test cases associated with the same partition are similar in terms of the execution behaviors aiming at achieving highly homogeneous partitions. However, we argue that input-domain partitioning may not fully satisfy this objective, as illustrated in the following example.

Example (Resource Allocation). Suppose that there are m projects, each of which generates a revenue of r; with a manpower requirement of pi (i = 1, 2, …, m), and n departments, each of which has ej (j = 1, 2, …, n) employees. A program Res attempts to assign projects to departments such that (a) each project is either assigned to one department or discarded, (b) the total manpower required from each department does not exceed its ej, and (c) the total revenue of all the assigned projects is maximized. The input for Res includes three sets of integers: two m-tuples R = (r1, r2, …, rm) and P = (p1, p2, …, pm), and one n-tuple E = (e1, e2, …, en). ∀i, ri > 0, pi > 0, and ∀j, ej > 0. The output of Res is one m-tuple S = (s1, s2, …, sm), sj = j (where i = 1, 2, …, m and j = 1, 2, …, n) represents that the ith project should be assigned to the jth department, while sj = 0 means that the jth project is discarded.

The categories and choices for Res are shown in Table I. For Res, there are six categories in total, each of which is associated with three choices. However, it does not mean that there will be 36 = 729 possible complete test frames, because some combinations of choices are invalid according to the specification. A total of 234 complete test frames can be constructed using algorithms provided by CHOिC’LATE.

Let us look at a complete test frame {1b, 2a, 3a, 4a, 6b}. Both of the following test cases can be generated from it:
- TC#1: R = (129, 129), P = (55, 55), E = (182).
- TC#2: R = (61, 61), P = (97, 97), E = (114).

Because (55 + 55 < 182), the output of TC#1 is S = (1, 1), that is, both projects are assigned to the only department. The output of TC#2 is S = (1, 0) or S = (0, 1), that is, only one project is assigned while the other is discarded (because (97 + 97 > 114)). In other words, TC#1 and TC#2 trigger

* This project is supported by an ARC grant.
different output scenarios, even though they come from
the same complete test frame (that is, the same input partition).

<table>
<thead>
<tr>
<th>Categories</th>
<th>Associated Choices</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Number of projects (m)</td>
<td>1a. m = 1, 1b. m = 2, 1c. m ≥ 3</td>
</tr>
<tr>
<td>2. Number of departments (n)</td>
<td>2a. n = 1, 2b. n = 2, 2c. n ≥ 3</td>
</tr>
<tr>
<td>3. Revenue of project (r)</td>
<td>3a. ∀ pair of i and j, r_i = r_j, 3b. ∀ pair of i and j, r_i ≠ r_j</td>
</tr>
<tr>
<td>4. Manpower for project (p)</td>
<td>4a. ∀ pair of i and j, p_i = p_j, 4b. ∀ pair of i and j, p_i ≠ p_j</td>
</tr>
<tr>
<td>5. Number of employers in department (e)</td>
<td>5a. ∀ pair of i and j, e_i = e_j, 5b. ∀ pair of i and j, e_i ≠ e_j</td>
</tr>
<tr>
<td>6. Relation between p_i and e_i</td>
<td>6a. ∀ i, p_i ≤ max (e_i, 6, e_i), 6b. ∀ i, p_i ≤ min (e_i, 6, e_i)</td>
</tr>
</tbody>
</table>

The above observation clearly shows that, although partitioning the input domain by existing methodologies (for example, CHOC’LATE) ensures the homogeneity in terms of the selected input aspects, some test cases from the same partition may still be heterogeneous with respect to output scenarios. Intuitively speaking, to maximize testing effectiveness, each partition should be as homogeneous as possible, and one good way to do this is to have fine-grained partitions that are not only related to input parameters but also corresponding to output scenarios.

IV. ENHANCING CHOC’LATE BY OUTPUT VARIATION

We suggest that, on top of the “traditional” partitioning of the input domain, a partition testing method should also consider the variation in program outputs, with a view to fine-tuning the test case generation process. Here, we propose an enhanced method, namely CHOC’LATE with IO-different output scenarios, even though they come from the same complete test frame (that is, the same input partition).

For clarity, those complete test frames containing 1-choices only are called I-based complete test frames (abbreviated as CTF_I). For Res, a total of 607 CTF_I can be generated. Table III shows the relevant statistics for Res, confirming that the situation of having a CTF_I associated with multiple CTF_O is very common.

<table>
<thead>
<tr>
<th>O-categories</th>
<th>O-choices</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Number of selected projects</td>
<td>1a. No project is assigned, 1b. All projects are assigned, 1c. Only some projects are assigned</td>
</tr>
<tr>
<td>2. Number of departments with projects assigned</td>
<td>2a. No department is assigned any project, 2b. All departments are assigned project(s), 2c. Only some departments are assigned project(s)</td>
</tr>
</tbody>
</table>

In CHOC’LATE-DIP over CHOC’LATE.

V. DISCUSSION AND CONCLUSION

The traditional approach to partitioning the input domain may not be sufficiently strong to ensure similar execution behaviors for the same resultant partitions. In this paper, we propose that output scenarios should also be explicitly considered for any partition testing method to improve the homogeneity of the input partitions, which, in turn, is the key factor for high fault-detection effectiveness. Such improvement is not only restricted to the testing method (CHOC’LATE) under this study, but can be generally used for enhancing many other partition testing techniques.

Due to the page limit, we only used one real-life system for the illustration and case study. A larger scale empirical study is our next step to investigate how to improve various partition testing methods on different types of systems.

REFERENCES