Metamorphic Testing of Concurrent Programs

August 31, 2017

1 Testing of Concurrent Programs

Testing of a concurrent program, e.g., a multi-threaded program, is still a challenging task due to its nondeterministic executions, among other testing issues including test case and oracle generation.

A concurrent program typically consists of multiple threads that cooperate to fulfill a functional assignment with high efficiency, e.g., computing, sorting, or searching a larger amount of data, especially for the data that may increase or decrease dynamically. Then, the oracle problem arises as it could be very expensive to verify the output results, if not impossible.

2 Metamorphic Testing of Concurrent Programs

We aim at investigating the applicability of metamorphic testing to concurrent programs. Metamorphic testing alleviates the oracle problem by exploiting the properties of the functional requirement or user expectation, which are identified as metamorphic relations, to evaluate the output results.

2.1 Metamorphic Relations

Consider a function $\textsc{top}(V, n)$ that returns in the ascending order the minimal n values of a list V, i.e., $\textsc{top}(V, n) = [y_1, \ldots, y_n]$ with $y_1 < \cdots < y_n$. V may contain duplicate elements.

Let VW be the concatenation of lists V and W, and $V^k = \underbrace{V \cdots V}_{k}$. Let $head(V)$ denote the first element of list V, and $tail(V)$ the list resulted by removing the first element of list V. That is, $V = head(V) :: tail(V)$. The higher-order function (map f V) applies the given function f to each element of list V, returning a list of the results in the same order, i.e., (map f V) $= f(head(V))$::(map f $tail(V)$). Then, metamorphic relations of $\textsc{top}(V, n)$ can be designed as follows.

2.1.1 Permutation

MR-1 $\textsc{top}(V', n) = \textsc{top}(V, n)$ for any permuation V' of V.

MR-2 (Commutative Law) $\text{TOP}(VW, n) = \text{TOP}(WV, n)$.

Permutation can be combined with other metamorphic relations.

2.1.2 Insertion

MR-3 $\text{TOP}(V^k, n) = \text{TOP}(V, n)$ for any $k > 1$.

MR-4 $\text{TOP}(VV', n) = \text{TOP}(V, n)$ for any $V' \subseteq V$.

MR-5 $\text{TOP}(V[y], n)$=$\text{TOP}(V, n)$ for each $y \in \text{TOP}(V, n)$.

MR-6 $\text{TOP}(VV_1 \cdots V_k, n) = \text{TOP}(V, n)$ for any $k \geq 1$, where $V_i \subseteq \text{TOP}(V, n)$ for every $1 \leq i \leq k$.

MR-7 $\text{TOP}(V[y], n) = [y_1, \ldots, y_i, y, y_{i+1}, \ldots, y_{n-1}]$ if $y \notin \text{TOP}(V, n)$ and there exists $1 \leq i < n$ such that $y_i < y < y_{i+1}$. Recall that $\text{TOP}(V, n) = [y_1, \ldots, y_n]$ with $y_1 < \cdots < y_n$.

MR-8 $\text{TOP}(V[y'_1, \ldots, y'_k], n) = [y'_1, \ldots, y'_k, y_1, \ldots, y_{n-k}]$, where $k \leq n$ and $y'_1 < \cdots < y'_k < y_1$.

MR-9 $\text{TOP}(V[y''_1, \ldots, y''_k], n) = \text{TOP}(V, n)$, where $y_n < y''_1 \leq \cdots \leq y''_k$.

MR-10 $\text{TOP}(VW, n) = \text{TOP}(V, n)$ for any permutation W of $\text{TOP}(V, n)$, especially $W = [y_1, \ldots, y_n]$ or $W = [y_n, \ldots, y_1]$.

MR-11 $\text{TOP}(WV, n) = \text{TOP}(V, n)$ for any permutation W of $\text{TOP}(V, n)$, especially $W = [y_1, \ldots, y_n]$ or $W = [y_n, \ldots, y_1]$.

2.1.3 Deletion

MR-12 Let $X = [x_1, \ldots, x_k]$ for arbitrary $k \geq 1$ elements $x_1, \ldots, x_k$, $V' = V \backslash X$, $X' = X \cap \text{TOP}(V, n)$.

- Then, $\text{TOP}(V, n) \backslash \text{TOP}(V', n) \subseteq X'$
- Specially, if $X' = \emptyset$, $\text{TOP}(V', n) = \text{TOP}(V, n)$;

where $V \backslash X$ removes from V only one occurrence of each element in X (if any).

Remark. A major concern with *MR*-12 is that the construction of the follow-up V' may be expensive. □

2.1.4 Replacement

MR-13 TOP((map f V),n)=(map f TOP(V,n)). For example, $f(x) = x + c$ for any constant c.

2.1.5 Splitting

Suppose $V = V_1V_2$. Then,

MR-14 $\text{TOP}(V, n) \subseteq \text{TOP}(V_1, n) \cup \text{TOP}(V_2, n)$.

MR-15 $\text{TOP}(V_1, n) \cap \text{TOP}(V_2, n) \subseteq \text{TOP}(V, 2n)$.

MR-16 $\text{TOP}(V_1, n) \cap \text{TOP}(V_2, m) \subseteq \text{TOP}(V, n+m)$. m在（1-9）之间取一个数，在（11-20）之间取一个数

MR-17 $\text{TOP}(V, n) = \text{TOP}(\text{TOP}(V_1, n)\text{TOP}(V_2, n), n)$.

2.1.6 Sublist

MR-18 $\text{TOP}(V, n)$ is a prefix of $\text{TOP}(V, m)$ if $n < m$, especially $n + 1 = m$. m的值取11或者取12-19之间的数值

MR-19 $\text{TOP}(V, n)$ is an extension of $\text{TOP}(V, m)$ if $n > m$, especially $n = m + 1$. m的值取9或者1-8

Remark. If the number of values contained in V is less than n, then $\text{TOP}(V, n)$ shall return all the values in V in the ascending order. We need to check whether the above MRs still hold. □

2.2 Roadmap

A concurrent program that implements the function $\text{TOP}(V, n)$ may use various concurrent priority queue classes, e.g., array-based, tree-based, heap-based or skiplist-based.

- Mutate the concurrent program for fault injection.
- Generate source test inputs.
- Generate follow-up test inputs based on the above metamorphic relations. [1]
- Run the mutants with the source and follow-up test inputs.
- Evaluate the output results of the source and follow-up test inputs against the corresponding metamorphic relation or composition of metamorphic relations.

---

[1] Later we can test the composition of these metamorphic relations, e.g., *MR*-3/*MR*-10. For example,

$$V = [y_1, \ldots, y_n]^k [y_{j_1}, \ldots, y_{j_p}] \quad (1)$$
$$V' = V[y_i']^q \quad (2)$$
$$V'' = V' \underbrace{\backslash[y_i'] \ldots \backslash[y_i']}_{q} \quad (3)$$

where $[y_{j_1}, \ldots, y_{j_k}] \subseteq [y_1, \ldots, y_n], y_i < y_i' < y_{i+1}, 1 < i < n$ and any $k, p, q > 1$.

2.3 Future Work

- execution-based metamorphic relations for concurrent programs, based on the consistency conditions of concurrent programs?
- metamorphic fault localization and repair of concurrent programs