Advanced Functional Programming – Autumn 2020
Assignment 2 - Stack Permutations

Li Ju

27th November 2020

Algorithm and implementation
Algorithm used for the problem can be referenced from the site provided in the problem description[1]. The algorithm can be described as following:

1. Starting from an input queue, an output queue and an empty stack.
2. If the output queue is empty, return true
3. Check if the element on the top of the stack or the first element of input queue equals to the first element of the output queue. If true: remove both equal elements. If false, dequeue the first element of the input queue and push it to the stack - if the input queue is empty, return false.
4. Go to step 2

In the implementation:
1). In Haskell, it is easier to match the first element of a list. However, given queues are supposed to pop the last element first. Therefore, function perm reverse input queues and feed reversed queues to function permS, which is the function doing the main computation. More over, an empty stack is also attached to permS. Therefore, Each time perm(input, output) is called, it will call permS(reverse(input), reverse(output), []), recursively do the computing and return the value. Only function perm is exported.

Property-based Testing
For property-based testing, 3 properties of the function are chosen: 1). all input queues containing 231 permutation patterns cannot be stack-ordered[2], 2). a list should be always stack permutable with itself, and 3). computation for lists whose length is less than 1000000 should be done within 1 second.

For the first property, the test function named prop_unsortable is created. The test function will generate an ordered list as the output_queue, then shuffle the output_queue till it contains at least 1 231 permutation patterns as the input_queue. The input_queue with 231 patterns will not be a stack permutation of the ordered queue. Therefore, the results of perm(input_queue, output_queue) should always be false, which will be checked by function prop_unsortable.

For the second property, the test function named prop_self is created. The function will generate random integers and call perm with the list as both arguments.

For the third property, the test function named prop_eff is created. A random list with non-repeative elements and its reverse will be the arguments to call function perm, if the length of the random list is less than 1000000. The test will fail only if the time cost is larger than 1000000 microsecond.

References

[1] Suprotik Dey. "Stack Permutations (Check if an array is stack permutation of other)". Geeksforgeeks, 2019, `https://www.geeksforgeeks.org/stack-permutations-check-if-an-array-is-stack-permutation-of-other/`

[2] Both Neou, Romeo Rizzi, Stéphane Vialette. "Permutation Pattern matching in (213, 231)-avoiding permutations". Discrete Mathematics and Theoretical Computer Science, DMTCS, 2017, 18 (2), pp.14.1-22. ffhal-01219299v6