Minimum Testing Requirements

James Wilcox and Kevin Zatloukal

May 2023

Our first rules supersede all others:

- **Exhaustive Testing**: for functions with at most 20 *allowed* inputs, test every input
- **Disallowed inputs**, either by the types or the specification, do not require testing
- Test each function individually. Assume the other functions it calls work properly

If a function has more than 20 allowed inputs, then we split them up into "subdomains" (subsets) using rules described below and then choose multiple test inputs for each subdomain, as described in the next section.

Testing a Subdomain

The following rules must be followed when choosing test inputs for an individual subdomain:

- Every subdomain containing at least 2 inputs must have **at least 2 test cases**.[1]
- Every **boundary case** must be tested along with at least one non-boundary case.[2]

Splitting Into Subdomains

For the simplest code, containing no loops or recursive calls chains[3], we use the following rule:

- Separate inputs that take **different paths** through the code into separate subdomains.

For functions with recursive calls, we split inputs based on the **maximum depth** of recursion made during the call. Specifically, maximum depth 0, 1, and 2+ are split into their own subdomains. We then split those inputs into smaller subdomains as follows:

- For inputs that make no recursive calls, inputs that take different paths are in separate subdomains.
- For inputs that make 1 recursive call, inputs that take different paths are in separate subdomains.[4]
- For inputs that make 2+ recursive calls, we focus on the outermost two calls in the call sequence. Inputs that take different paths in the outermost call or the first recursive call are in separate subdomains.

[1] One input subdomains require only one test.

[2] In general, if the subdomain has $n \geq 1$ boundaries, then the subdomain requires $n + 1$ test cases.

[3] Any sequence of calls that has a call to our function while still executing an existing call to our function counts as recursion, even if the function does not call itself directly.

[4] Note that the path includes the code executed in both the outer call and the inner call.