STiDC'07: Exercise 3

October 22, 2007 (updated on October 30, 2007)

1 Problem

A *splitter* is a shared object that has only one operation, called *splitter*, that can return *stop*, *left* or *right*. Every splitter object ensures the following:

1. If a single process executes *splitter*, then the process is returned *stop*;
2. If two or more processes execute *splitter*, then not all of them get the same output value; and
3. At most one process is returned *stop*.

Your task is to implement a wait-free, atomic splitter object using *only* atomic (multi-valued, MRMW) registers.

2 Solution

We use two registers:

- P (multi-valued), and
- S (binary, initialized to *false*)

Code for process p_i:

```
upon $splitter_i$
    $P \leftarrow i$
    if $S$ then return "right"
    $S \leftarrow true$
    if $P = i$ then return "stop"
    return "left"
```

Note: The implementation is atomic (linearizable). This means that if a process p_i crashes while executing the *splitter* operation, we can assume that either (1) the operation did not take place at all (no linearization point), or (2) the operation has completed with some (arbitrarily chosen) return value (linearization point can be anywhere after the operation starts).