Exercise 6

1 Emulating a \((1, N)\) Register from \((1, 1)\) Registers

Consider the implementation in Algorithm 1 of a \((1, N)\) register, instance \(onr\), from an array of \(N\) instances of so-called base registers. This algorithm sends no messages explicitly, it merely reduces one abstraction to another one. It is an example of an algorithm in the so-called shared memory-model.

The unique writer process of the \((1, N)\) register is \(p\). The base registers are \((1, 1)\) registers, denoted \(br.q\) for \(q \in \Pi\), such that only process \(p\) may write to instance \(br.q\) and only process \(q\) may read from it. (Recall that \(self\) denotes the process executing the algorithm. The consistency property of the registers, whether they are safe, regular, or atomic, is specified later.)

Algorithm 1: Multi-Reader Emulation

**Implements:**

\((1, N)\)-Register, instance \(onr\). \hspace{1cm} // the writer is \(p\)

**Uses:**

\((1, 1)\)-Register (multiple instances).

**upon event** \(\langle onr, \text{Init} \rangle\) **do**

\(\text{writeset} := \emptyset;\)

forall \(q \in \Pi\) **do**

Initialize a new instance \(br.q\) of \((1, 1)\)-Register with writer \(p\) and reader \(q\);

**upon event** \(\langle onr, \text{Read} \rangle\) **do**

trigger \(\langle br.self, \text{Read} \rangle;\)

**upon event** \(\langle br.self, \text{ReadReturn} \mid v \rangle\) **do**

trigger \(\langle onr, \text{ReadReturn} \mid v \rangle;\)

**upon event** \(\langle onr, \text{Write} \mid v \rangle\) **do**

forall \(q \in \Pi\) **do**

trigger \(\langle br.q, \text{Write} \mid v \rangle;\)

**upon event** \(\langle br.q, \text{WriteReturn} \rangle\) **do**

\(\text{writeset} := \text{writeset} \cup \{q\};\)

if \(\text{writeset} = \Pi\) then

\(\text{writeset} := \emptyset;\)

trigger \(\langle onr, \text{WriteReturn} \rangle;\)

// only the writer \(p\)
Answer these questions and justify your answers:

(a) Let the array $br.q$ for $q \in \Pi$ be safe binary $(1, 1)$-registers. Show that the emulation produces a safe binary $(1, N)$-register instance onr.

(b) If we replace the $N$ safe registers $br.q$ for $q \in \Pi$ with an array of regular binary $(1, 1)$-registers (i.e., registers that only store one bit), does the algorithm implement a regular binary $(1, N)$-register?

(c) If we replace the $N$ safe registers $br.q$ for $q \in \Pi$ with an array of regular multi-valued $(1, 1)$-registers, does the algorithm implement a regular multi-valued $(1, N)$-register?

2 Perfect Failure Detector from Regular Register

Consider a system with $N$ processes, where up to $N - 1$ processes may fail by crashing and have synchronized clocks available. Assume you are given an algorithm that implements a $(1, N)$ regular register abstraction. How can this abstraction be used to implement a perfect failure detector (Module 2.6 in [CGR11])?

Hint: Use $N$ register instances.