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*. // the writer is p

Uses:
$(1, 1)$-Register (multiple instances).

upon event $\langle$ *onr*, *Init* $\rangle$ **do**
  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*, *Read* $\rangle$ **do**
  trigger $\langle$ *br.self*, *Read* $\rangle$;

upon event $\langle$ *br.self*, *ReadReturn* | v $\rangle$ **do**
  trigger $\langle$ *onr*, *ReadReturn* | v $\rangle$;

upon event $\langle$ *onr*, *Write* | v $\rangle$ **do** // only the writer p
  forall $q \in \Pi$ **do**
    trigger $\langle$ $br.q$, *Write* | v) $\rangle$;

upon event $\langle$ $br.q$, *WriteReturn* $\rangle$ **do** // only the writer p
  writeset := *writeset* $\cup \{q\}$;
  if *writeset* $= \Pi$ **then**
    writeset := $\emptyset$;
    trigger $\langle$ *onr*, *WriteReturn* $\rangle$;

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.