Concurrent Computing December 16, 2024

Exercise 11

Problem 1.

Consider the atomic *commit-adopt* object, which has the following specification. Every process p proposes an input value v to such an object and obtains an output, which consists of a pair (dec, val); dec can be either *commit* or *adopt*. The following properties are satisfied:

- **Validity:** If a process obtains output (commit, v) or (adopt, v), then v was proposed by some process.
- **Commitment:** If every process proposes the same value, then no process may output (adopt, v) for any value v.
- **Agreement:** If a process p outputs (commit, v) and a process q outputs (commit, v') or (adopt, v'), then $v = v'$.
- **Termination:** Every correct process eventually obtains an output.

Consider the following implementation of an atomic *commit-adopt* object from atomic wait-free snapshot objects and atomic MRMW registers:

- Using two shared snapshot objects: S_1 and S_2 of size n, initialized to $(\perp, \perp, \ldots, \perp)$;
- Using two local arrays of registers: a_i and b_i of size n.

The implementation is as follows:

```
propose(v)
    S_1.update$i$, $v$;

    a_i := S_1.snapshot();
    if every non-⊥ value in a_i is v then
        x := (true, v);
    else
        v := max(a_i); // max(arr) returns the greatest non-⊥ element in array arr
        x := (false, v);

    S_2.update(i, x);

    b_i := S_2.snapshot();
    if every non-⊥ value in b_i is equal to (true, v) then
        return (commit, v);
    if some value in b_i is equal to (true, val) for some val then
        return (adopt, val);
    return (adopt, v);
```

Is the above implementation correct (does it satisfy the *commit-adopt* properties)? Justify your answer.