STiDC'06: Exercise 4

November 28, 2006

Write an algorithm (in $^+$CAL) that implements a *strong counter* using atomic registers and compare-and-swap objects. Devise a set of assertions and/or invariants (in $^+$CAL/TLA$^+$) that check for atomicity of the implemented strong counter, for a given number of processes (N) and a given number of invocations of operation *inc* per process (K).

A strong counter is a shared object that maintains a single variable c, initialized to 0, and provides a single operation *inc* with the following sequential specification:

```
operation inc()
  c := c + 1
  return c
end
```

A compare-and-swap object is a shared object that maintains a single variable v, initialized to $\perp$, and provides a single operation *CAS* with the following sequential specification:

```
operation CAS(oldVal, newVal)
  if v = oldVal then v := newVal
  return v
end
```