STiDC'08: Exercise 6

January 16, 2009

Write an algorithm that implements a *fetch-and-increment* object using atomic registers and compare-and-swap objects.

Reminder: Fetch-and-increment is a shared object that maintains a single variable c, initialized to 0, and provides a single operation *fetch&inc* with the following sequential specification:

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

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

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

Solution. Here is an example algorithm that implements a fetch-and-increment object using: (1) a single compare-and-swap object C (initialized to $\langle -1, \ldots, -1 \rangle$), and (2) array R of N atomic registers (each initialized to -2). The local variable (array) $last_i$ is initialized to $\langle -1, \ldots, -1 \rangle$ at every process p_i.

upon *fetch&inc()$_i$* **do**
  $R[i] \leftarrow last_i[i]$
  repeat
      for $k \leftarrow 1$ **to** N **do** $r[k] \leftarrow R[k]$
      $m \leftarrow \max_k(r[k]) + 1$
      $new \leftarrow last_i$
      for $k \leftarrow 1$ **to** N **do**
          if $r[k] = last_i[k]$ **then**
              $new[k] \leftarrow m$
              $m \leftarrow m + 1$
      $v \leftarrow C.CAS(last_i, new)$
      if $v = last_i$ **then** $last_i \leftarrow new$
      else $last_i \leftarrow v$
  until $last_i[i] > R[i]$
  return $last_i[i]$