MODULE *BakeryDistributed*

This is the *PlusCal* specification of the distributed bakery algorithm in the paper

Deconstructing the Bakery to Build a Distributed State Machine

We assume here that you have read the *BakeryDeconstructed* specification, whose comments explain the structure of this *PlusCal* translation of the pseudo-code in the paper.

The statements in gray in the paper's pseudo-code, which involve the unnecessary variable *localCh* may be commented or uncommented for checking purposes.

EXTENDS *Data*, *Sequences*

We let *ack* denote an arbitrary value that is not an integer. Although this algorithm does not use value *qm* , we make the values different in order to avoid accidental equality.

$ack \triangleq \text{CHOOSE } v : v \notin Nat \cup \{qm\}$

```
***************************************************************************
--algorithm Dist{
  variables number = [i ∈ Procs ↦ 0],
            localNum = [i ∈ Procs ↦ [j ∈ OtherProcs(i) ↦ 0]],
            localCh  = [i ∈ Procs ↦ [j ∈ OtherProcs(i) ↦ 0]],    ERASABLE
            ackRcvd  = [i ∈ Procs ↦ [j ∈ OtherProcs(i) ↦ 0]],
            q = [i ∈ Procs ↦ [j ∈ OtherProcs(i) ↦ ⟨⟩]]

  fair process (main ∈ ProcIds){
    ncs:- while (TRUE){
            skip ;  noncritical section
        M: await ∀ p ∈ SubProcsOf(self[1]) : pc[p] = "L0" ;
           with (v ∈ {n ∈ Nat \ {0} :
                          ∀ j ∈ OtherProcs(self[1]) :
                             n > localNum[self[1]][j]}){
             number[self[1]] := v ;
             q[self[1]] := [j ∈ OtherProcs(self[1])
                              ↦ Append(q[self[1]][j], v)]
           } ;
        L: await ∀ p ∈ SubProcsOf(self[1]) : pc[p] = "ch" ;
       cs: skip ;  critical section
        P: ackRcvd[self[1]] := [j ∈ OtherProcs(self[1]) ↦ 0] ;
           number[self[1]] := 0 ;
           q[self[1]] := [j ∈ OtherProcs(self[1])
                            ↦ Append(q[self[1]][j], 0)]
         }
  }

  fair process (sub ∈ SubProcs){
    ch: while (TRUE){
          await pc[⟨self[1]⟩] = "M" ;
          localCh[self[2]][self[1]] := 1 ;    ERASABLE
```

```
      L0:  await pc[⟨self[1]⟩] = "L" ;
           await ackRcvd[self[1]][self[2]] = 1 ;
           localCh[self[2]][self[1]] := 0 ;         ERASABLE
      L2:  await localCh[self[1]][self[2]] = 0 ;    ERASABLE
      L3:  await  ∨ localNum[self[1]][self[2]] = 0
                  ∨ ⟨number[self[1]], self[1]⟩ ≪
                       ⟨localNum[self[1]][self[2]], self[2]⟩
        }
  }

  fair process (msg ∈ MsgProcs){
    wr: while (TRUE){
          await q[self[2]][self[1]] ≠ ⟨⟩ ;
          with (v = Head(q[self[2]][self[1]])){
            if (v   = ack){ackRcvd[self[1]][self[2]] := 1}
            else {localNum[self[1]][self[2]] := v} ;
            if (v ∈ {0, ack}){
               q[self[2]][self[1]] := Tail(q[self[2]][self[1]])}
            else {q[self[2]][self[1]] := Tail(q[self[2]][self[1]]) ||
                    q[self[1]][self[2]] := Append(q[self[1]][self[2]], ack)}
            }
        }
  }
}
*************************************************************************
```

```
 BEGIN TRANSLATION (chksum(pcal) = "d4d60f14" ∧ chksum(tla) = "8b3daef")
VARIABLES number, localNum, localCh, ackRcvd, q, pc

vars ≜ ⟨number, localNum, localCh, ackRcvd, q, pc⟩

ProcSet ≜ (ProcIds) ∪ (SubProcs) ∪ (MsgProcs)

Init ≜  Global variables
         ∧ number = [i ∈ Procs ↦ 0]
         ∧ localNum = [i ∈ Procs ↦ [j ∈ OtherProcs(i) ↦ 0]]
         ∧ localCh  = [i ∈ Procs ↦ [j ∈ OtherProcs(i) ↦ 0]]
         ∧ ackRcvd = [i ∈ Procs ↦ [j ∈ OtherProcs(i) ↦ 0]]
         ∧ q = [i ∈ Procs ↦ [j ∈ OtherProcs(i) ↦ ⟨⟩]]
         ∧ pc = [self ∈ ProcSet ↦ CASE self ∈ ProcIds → "ncs"
                                    □   self ∈ SubProcs → "ch"
                                    □   self ∈ MsgProcs → "wr"]

ncs(self) ≜  ∧ pc[self] = "ncs"
             ∧ TRUE
             ∧ pc' = [pc EXCEPT ![self] = "M"]
             ∧ UNCHANGED ⟨number, localNum, localCh, ackRcvd, q⟩
```

$$M(\mathit{self}) \triangleq \begin{array}[t]{l} \land pc[\mathit{self}] = \text{``M''} \\ \land \forall p \in \mathit{SubProcsOf}(\mathit{self}[1]) : pc[p] = \text{``L0''} \\ \land \exists v \in \{n \in \mathit{Nat} \setminus \{0\} : \\ \qquad\qquad \forall j \in \mathit{OtherProcs}(\mathit{self}[1]) \quad : \\ \qquad\qquad\quad n > \mathit{localNum}[\mathit{self}[1]][j]\} : \\ \qquad \land \mathit{number}' = [\mathit{number} \textsc{ except } ![\mathit{self}[1]] = v] \\ \qquad \land q' = [q \textsc{ except } ![\mathit{self}[1]] = [j \in \mathit{OtherProcs}(\mathit{self}[1]) \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad \mapsto \mathit{Append}(q[\mathit{self}[1]][j],\ v)]] \\ \land pc' = [pc \textsc{ except } ![\mathit{self}] = \text{``L''}] \\ \land \textsc{unchanged } \langle \mathit{localNum},\ \mathit{localCh},\ \mathit{ackRcvd} \rangle \end{array}$$

$$L(\mathit{self}) \triangleq \begin{array}[t]{l} \land pc[\mathit{self}] = \text{``L''} \\ \land \forall p \in \mathit{SubProcsOf}(\mathit{self}[1]) : pc[p] = \text{``ch''} \\ \land pc' = [pc \textsc{ except } ![\mathit{self}] = \text{``cs''}] \\ \land \textsc{unchanged } \langle \mathit{number},\ \mathit{localNum},\ \mathit{localCh},\ \mathit{ackRcvd},\ q \rangle \end{array}$$

$$cs(\mathit{self}) \triangleq \begin{array}[t]{l} \land pc[\mathit{self}] = \text{``cs''} \\ \land \textsc{true} \\ \land pc' = [pc \textsc{ except } ![\mathit{self}] = \text{``P''}] \\ \land \textsc{unchanged } \langle \mathit{number},\ \mathit{localNum},\ \mathit{localCh},\ \mathit{ackRcvd},\ q \rangle \end{array}$$

$$P(\mathit{self}) \triangleq \begin{array}[t]{l} \land pc[\mathit{self}] \quad = \text{``P''} \\ \land \mathit{ackRcvd}' = [\mathit{ackRcvd} \textsc{ except } ![\mathit{self}[1]] = [j \in \mathit{OtherProcs}(\mathit{self}[1]) \mapsto 0]] \\ \land \mathit{number}' = [\mathit{number} \textsc{ except } ![\mathit{self}[1]] = 0] \\ \land q' = [q \textsc{ except } ![\mathit{self}[1]] = [j \in \mathit{OtherProcs}(\mathit{self}[1]) \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad \mapsto \mathit{Append}(q[\mathit{self}[1]][j],\ 0)]] \\ \land pc' = [pc \textsc{ except } ![\mathit{self}] = \text{``ncs''}] \\ \land \textsc{unchanged } \langle \mathit{localNum},\ \mathit{localCh} \rangle \end{array}$$

$$\mathit{main}(\mathit{self}) \triangleq \mathit{ncs}(\mathit{self}) \lor M(\mathit{self}) \lor L(\mathit{self}) \lor cs(\mathit{self}) \lor P(\mathit{self})$$

$$ch(\mathit{self}) \triangleq \begin{array}[t]{l} \land pc[\mathit{self}] = \text{``ch''} \\ \land pc[\langle \mathit{self}[1] \rangle] = \text{``M''} \\ \land \mathit{localCh}' = [\mathit{localCh} \textsc{ except } ![\mathit{self}[2]][\mathit{self}[1]] = 1] \\ \land pc' = [pc \textsc{ except } ![\mathit{self}] = \text{``L0''}] \\ \land \textsc{unchanged } \langle \mathit{number},\ \mathit{localNum},\ \mathit{ackRcvd},\ q \rangle \end{array}$$

$$L0(\mathit{self}) \triangleq \begin{array}[t]{l} \land pc[\mathit{self}] = \text{``L0''} \\ \land pc[\langle \mathit{self}[1] \rangle] = \text{``L''} \\ \land \mathit{ackRcvd}[\mathit{self}[1]][\mathit{self}[2]] = 1 \\ \land \mathit{localCh}' = [\mathit{localCh} \textsc{ except } ![\mathit{self}[2]][\mathit{self}[1]] = 0] \\ \land pc' = [pc \textsc{ except } ![\mathit{self}] = \text{``L2''}] \\ \land \textsc{unchanged } \langle \mathit{number},\ \mathit{localNum},\ \mathit{ackRcvd},\ q \rangle \end{array}$$

$$L2(\mathit{self}) \triangleq \begin{array}[t]{l} \land pc[\mathit{self}] = \text{``L2''} \\ \land \mathit{localCh}[\mathit{self}[1]][\mathit{self}[2]] = 0 \\ \land pc' = [pc \textsc{ except } ![\mathit{self}] = \text{``L3''}] \end{array}$$

```
                  $\land \textsc{unchanged}\ \langle number, localNum, localCh, ackRcvd, q\rangle$

$L3(self) \triangleq$ $\land pc[self] = \text{“L3”}$
                  $\land \lor localNum[self[1]][self[2]] = 0$
                     $\lor \langle number[self[1]], self[1]\rangle \ll$
                         $\langle localNum[self[1]][self[2]], self[2]\rangle$
                  $\land pc' = [pc\ \textsc{except}\ ![self] = \text{“ch”}]$
                  $\land \textsc{unchanged}\ \langle number, localNum, localCh, ackRcvd, q\rangle$

$sub(self) \triangleq ch(self) \lor L0(self) \lor L2(self) \lor L3(self)$

$wr(self) \triangleq$ $\land pc[self] = \text{“wr”}$
                  $\land q[self[2]][self[1]] \neq \langle\rangle$
                  $\land \textsc{let}\ v \triangleq Head(q[self[2]][self[1]])\textsc{in}$
                       $\land \textsc{if}\ v = ack$
                              $\textsc{then}\ \land ackRcvd' = [ackRcvd\ \textsc{except}\ ![self[1]][self[2]] = 1]$
                                      $\land \textsc{unchanged}\ localNum$
                              $\textsc{else}\ \land localNum' = [localNum\ \textsc{except}\ ![self[1]][self[2]] = v]$
                                      $\land \textsc{unchanged}\ ackRcvd$
                       $\land \textsc{if}\ v \in \{0, ack\}$
                              $\textsc{then}\ \land q' = [q\ \textsc{except}\ ![self[2]][self[1]] = Tail(q[self[2]][self[1]])]$
                              $\textsc{else}\ \land q' = [q\ \textsc{except}\ ![self[2]][self[1]] = Tail(q[self[2]][self[1]]),$
                                                          $![self[1]][self[2]] = Append(q[self[1]][self[2]], ack)]$
                  $\land pc' = [pc\ \textsc{except}\ ![self] = \text{“wr”}]$
                  $\land \textsc{unchanged}\ \langle number, localCh\rangle$

$msg(self) \triangleq wr(self)$

$Next \triangleq (\exists\, self \in ProcIds : main(self))$
           $\lor (\exists\, self \in SubProcs : sub(self))$
           $\lor (\exists\, self \in MsgProcs : msg(self))$

$Spec \triangleq$ $\land Init \land \Box[Next]_{vars}$
          $\land \forall\, self \in ProcIds : \mathrm{WF}_{vars}((pc[self] \neq \text{“ncs”}) \land main(self))$
          $\land \forall\, self \in SubProcs : \mathrm{WF}_{vars}(sub(self))$
          $\land \forall\, self \in MsgProcs : \mathrm{WF}_{vars}(msg(self))$

 END TRANSLATION
```

```
$TypeOK \triangleq$ $\land number \in [Procs \to Nat]$
                  $\land \land \textsc{domain}\ localNum = Procs$
                     $\land \forall\, i \in Procs : localNum[i] \in [OtherProcs(i) \to Nat]$
          $\land \land \textsc{domain}\ localCh = Procs$
           $\land \forall\, i \in Procs : localCh[i] \in [OtherProcs(i) \to \{0, 1\}]$ ERASABLE
             $\land \land \textsc{domain}\ ackRcvd = Procs$
                $\land \forall\, i \in Procs : ackRcvd[i] \in [OtherProcs(i) \to \{0, 1\}]$
             $\land \land \textsc{domain}\ q = Procs$
```

$$
\begin{array}{ll}
 & \land\ \forall i \in Procs : q[i] \in [OtherProcs(i) \rightarrow Seq(Nat \cup \{ack\})] \\
\land & \land\ \text{DOMAIN}\ pc = ProcSet \\
 & \land\ \forall p \in ProcSet : \\
 & \qquad \text{CASE}\ p \in ProcIds \rightarrow pc[p] \in \{\text{“ncs”}, \text{“M”}, \text{“L”}, \text{“cs”}, \text{“P”}\} \\
 & \qquad\quad \Box \quad p \in SubProcs \rightarrow pc[p] \in \{\text{“ch”}, \text{“L0”}, \text{“L2”}, \text{“L3”}\} \\
 & \qquad\quad \Box \quad p \in MsgProcs \rightarrow pc[p] = \text{“wr”}
\end{array}
$$

$$MutualExclusion \triangleq \forall p, r \in ProcIds : (p \neq r) \Rightarrow (\{pc[p], pc[r]\} \neq \{\text{“cs”}\})$$

$$StarvationFree \triangleq \forall p \in ProcIds : (pc[p] = \text{“M”}) \leadsto (pc[p] = \text{“cs”})$$

$$TestMaxNum \triangleq 4$$

$$TestNat \triangleq 0 \,..\, (TestMaxNum + 1)$$

****

Version of 28 *Apr* 2021 Tests checking *MutualExclusion*, *StarvationFree*

$N = 2$, $TestMaxNum = 6$, 2,993 states (2590 states without *localCh*)

$N = 3$, $TestMaxNum = 3$, 1,714,288 states 2:32 + 1:22 on Azure

$N = 3$, $TestMaxNum = 4$, 5,071,345 states 7:05 + 3:12 on Azure

$N = 3$, $TestMaxNum = 5$, 12,071,392 states 16:05 + 7:58 on Azure (+ *TypeOK*)

****

\* Modification History
\* Last modified *Tue Sep* 07 16:33:12 *CEST* 2021 by *merz*
\* Last modified *Thu Sep* 02 11:56:50 *CEST* 2021 by *merz*
\* Last modified *Wed Apr* 28 18:11:29 *PDT* 2021 by *lamport*
\* Created *Tue Apr* 27 10:33:38 *PDT* 2021 by *lamport*