MODULE *Bakery*

The bakery algorithm originally appeared in:

Leslie *Lamport* A New Solution of *Dijkstra*'s Concurrent Programming Problem Communications of the *ACM* 17, 8 (August 1974), 453 – 455

The code for the algorithm given in that paper is :

```
begin integer j;
L1: choosing [i] := 1 ;
    number[i] := 1 + maximum (number[1],..., number[N]);
    choosing[i] := 0;
    for j = 1 step 1 until N do
       begin
         L2: if choosing[j] /= 0 then goto L2;
         L3: if number[j] /= 0 and (number [j], j) < (number[i],i)
               then goto L3;
       end;
    critical section;
    number[i] := 0;
    noncritical section;
    goto L1 ;
end
```

What makes the bakery algorithm interesting is that it doesn't assume that reading or writing a memory register is an atomic operation. Instead it assumes safe registers, which ensure only that a read that doesn't overlap a write obtains the current value of the register, but allows a read that overlaps a write to obtain any value of the correct type. This is modeled in TLA+ by letting the read be atomic but having a write operation perform a sequence of atomic writes of arbitrary type-correct values before atomically writing the desired value. (Only the shared registers $number[i]$ and $choosing[i]$ need be to be modeled in this way; operations to a process's local variables can be taken to be atomic.)

This *PlusCal* version of the Atomic *Bakery* algorithm is one in which variables whose initial values are not used are initialized to particular type-correct values. If the variables were left uninitialized, the *PlusCal* translation would initialize them to a particular unspecified value. This would complicate the proof because it would make the type-correctness invariant more complicated, but it would be more efficient to model check. We could write a version that is more elegant and easy to prove, but less efficient to model check, by initializing the variables to arbitrarily chosen type-correct values.

EXTENDS *Naturals*, *TLAPS*

We first declare N to be the number of processes, and we assume that N is a natural number.

CONSTANT N

ASSUME $N \in Nat$

We define *Procs* to be the set $\{1, 2, \ldots, N\}$ of processes.

$$Procs \triangleq 1 \,..\, N$$

$\prec$ is defined to be the lexicographical less-than relation on pairs of numbers.

$$a \prec b \triangleq \begin{array}{l} \lor\ a[1] < b[1] \\ \lor\ (a[1] = b[1]) \land (a[2] < b[2]) \end{array}$$

$**$ this is a comment containing the *PlusCal* code *

```
--algorithm Bakery
{ variables num = [i ∈ Procs ↦ 0], flag = [i ∈ Procs ↦ FALSE] ;
  fair process ( p ∈ Procs )
    variables unchecked = {}, max = 0, nxt = 1 ;
    { ncs:- while ( TRUE )
            { e1:  either { flag[self] := ¬flag[self] ;
                            goto e1 }
                   or     { flag[self] := TRUE ;
                            unchecked := Procs \ {self} ;
                            max := 0
                          } ;
              e2:  while ( unchecked ≠ {} )
                     { with ( i ∈ unchecked )
                         { unchecked := unchecked \ {i} ;
                           if ( num[i] > max ) { max := num[i] }
                         }
                     } ;
              e3:  either { with ( k ∈ Nat ) { num[self] := k } ;
                            goto e3 }
                   or     { with ( i ∈ {j ∈ Nat : j > max} )
                              { num[self] := i }
                          } ;
              e4:  either { flag[self] := ¬flag[self] ;
                            goto e4 }
                   or     { flag[self] := FALSE ;
                            unchecked := Procs \ {self}
                          } ;
              w1:  while ( unchecked ≠ {} )
                     {    with ( i ∈ unchecked ) { nxt := i } ;
                          await ¬flag[nxt] ;
                      w2: await  ∨ num[nxt] = 0
                                 ∨ ⟨num[self], self⟩ ≺ ⟨num[nxt], nxt⟩ ;
                          unchecked := unchecked \ {nxt} ;
                     } ;
              cs:  skip ;   the critical section;
              exit: either { with ( k ∈ Nat ) { num[self] := k } ;
                             goto exit }
                    or     { num[self] := 0 }
            }
    }
}
***    this ends the comment containg the pluscal code     *********
```

BEGIN TRANSLATION (this begins the translation of the *PlusCal* code)

VARIABLES num, $flag$, pc, $unchecked$, max, nxt

$$vars \triangleq \langle num,\ flag,\ pc,\ unchecked,\ max,\ nxt \rangle$$

$$ProcSet \triangleq (Procs)$$

$$\begin{array}{ll}
Init \triangleq & \text{Global variables} \\
& \land\ num = [i \in Procs \mapsto 0] \\
& \land\ flag = [i \in Procs \mapsto \text{FALSE}] \\
& \text{Process } p \\
& \land\ unchecked = [self \in Procs \mapsto \{\}] \\
& \land\ max = [self \in Procs \mapsto 0] \\
& \land\ nxt\ \ = [self \in Procs \mapsto 1] \\
& \land\ pc = [self \in ProcSet \mapsto \text{"ncs"}]
\end{array}$$

$$\begin{array}{ll}
ncs(self) \triangleq & \land\ pc[self] = \text{"ncs"} \\
& \land\ pc' = [pc \text{ EXCEPT } ![self] = \text{"e1"}] \\
& \land\ \text{UNCHANGED } \langle num,\ flag,\ unchecked,\ max,\ nxt \rangle
\end{array}$$

$$\begin{array}{ll}
e1(self) \triangleq & \land\ pc[self] = \text{"e1"} \\
& \land\ \lor\ \land\ flag' = [flag \text{ EXCEPT } ![self] = \neg flag[self]] \\
& \qquad\ \ \land\ pc' = [pc \text{ EXCEPT } ![self] = \text{"e1"}] \\
& \qquad\ \ \land\ \text{UNCHANGED } \langle unchecked,\ max \rangle \\
& \quad\ \lor\ \land\ flag' = [flag \text{ EXCEPT } ![self] = \text{TRUE}] \\
& \qquad\ \ \land\ unchecked' = [unchecked \text{ EXCEPT } ![self] = Procs \setminus \{self\}] \\
& \qquad\ \ \land\ max' = [max \text{ EXCEPT } ![self] = 0] \\
& \qquad\ \ \land\ pc' = [pc \text{ EXCEPT } ![self] = \text{"e2"}] \\
& \land\ \text{UNCHANGED } \langle num,\ nxt \rangle
\end{array}$$

$$\begin{array}{ll}
e2(self) \triangleq & \land\ pc[self] = \text{"e2"} \\
& \land\ \text{IF } unchecked[self] \neq \{\} \\
& \qquad \text{THEN } \land\ \exists\, i \in unchecked[self] : \\
& \qquad\qquad\qquad \land\ unchecked' = [unchecked \text{ EXCEPT } ![self] = unchecked[self] \setminus \{i\}] \\
& \qquad\qquad\qquad \land\ \text{IF } num[i] > max[self] \\
& \qquad\qquad\qquad\qquad \text{THEN } \land\ max' = [max \text{ EXCEPT } ![self] = num[i]] \\
& \qquad\qquad\qquad\qquad \text{ELSE } \ \land\ \text{TRUE} \\
& \qquad\qquad\qquad\qquad\qquad\ \ \land\ max' = max \\
& \qquad\qquad\quad \land\ pc' = [pc \text{ EXCEPT } ![self] = \text{"e2"}] \\
& \qquad \text{ELSE } \ \land\ pc' = [pc \text{ EXCEPT } ![self] = \text{"e3"}] \\
& \qquad\qquad\quad \land\ \text{UNCHANGED } \langle unchecked,\ max \rangle \\
& \land\ \text{UNCHANGED } \langle num,\ flag,\ nxt \rangle
\end{array}$$

$$\begin{array}{ll}
e3(self) \triangleq & \land\ pc[self] = \text{"e3"} \\
& \land\ \lor\ \land\ \exists\, k \in Nat : \\
& \qquad\qquad num' = [num \text{ EXCEPT } ![self] = k] \\
& \qquad\ \ \land\ pc' = [pc \text{ EXCEPT } ![self] = \text{"e3"}] \\
& \quad\ \lor\ \land\ \exists\, i \in \{j \in Nat : j > max[self]\} : \\
& \qquad\qquad num' = [num \text{ EXCEPT } ![self] = i] \\
& \qquad\ \ \land\ pc' = [pc \text{ EXCEPT } ![self] = \text{"e4"}]
\end{array}$$

$$
\begin{array}{ll}
 & \land \text{UNCHANGED } \langle \mathit{flag}, \mathit{unchecked}, \mathit{max}, \mathit{nxt} \rangle
\end{array}
$$

$$
\begin{array}{lll}
e4(\mathit{self}) & \triangleq & \land\, \mathit{pc}[\mathit{self}] = \text{“e4”} \\
 & & \land\, \lor\, \land\, \mathit{flag}' = [\mathit{flag} \text{ EXCEPT } ![\mathit{self}] = \neg \mathit{flag}[\mathit{self}]] \\
 & & \quad\;\;\, \land\, \mathit{pc}' = [\mathit{pc} \text{ EXCEPT } ![\mathit{self}] = \text{“e4”}] \\
 & & \quad\;\;\, \land\, \text{UNCHANGED } \mathit{unchecked} \\
 & & \quad \lor\, \land\, \mathit{flag}' = [\mathit{flag} \text{ EXCEPT } ![\mathit{self}] = \text{FALSE}] \\
 & & \quad\;\;\, \land\, \mathit{unchecked}' = [\mathit{unchecked} \text{ EXCEPT } ![\mathit{self}] = \mathit{Procs} \setminus \{\mathit{self}\}] \\
 & & \quad\;\;\, \land\, \mathit{pc}' = [\mathit{pc} \text{ EXCEPT } ![\mathit{self}] = \text{“w1”}] \\
 & & \land\, \text{UNCHANGED } \langle \mathit{num}, \mathit{max}, \mathit{nxt} \rangle
\end{array}
$$

$$
\begin{array}{lll}
w1(\mathit{self}) & \triangleq & \land\, \mathit{pc}[\mathit{self}] = \text{“w1”} \\
 & & \land\, \text{IF } \mathit{unchecked}[\mathit{self}] \neq \{\} \\
 & & \quad\quad \text{THEN } \land\, \exists\, i \in \mathit{unchecked}[\mathit{self}] : \\
 & & \quad\quad\quad\quad\quad\quad \mathit{nxt}' = [\mathit{nxt} \text{ EXCEPT } ![\mathit{self}] = i] \\
 & & \quad\quad\quad\quad\;\;\, \land\, \neg \mathit{flag}[\mathit{nxt}'[\mathit{self}]] \\
 & & \quad\quad\quad\quad\;\;\, \land\, \mathit{pc}' = [\mathit{pc} \text{ EXCEPT } ![\mathit{self}] = \text{“w2”}] \\
 & & \quad\quad \text{ELSE } \land\, \mathit{pc}' = [\mathit{pc} \text{ EXCEPT } ![\mathit{self}] = \text{“cs”}] \\
 & & \quad\quad\quad\quad\;\;\, \land\, \mathit{nxt}' = \mathit{nxt} \\
 & & \land\, \text{UNCHANGED } \langle \mathit{num}, \mathit{flag}, \mathit{unchecked}, \mathit{max} \rangle
\end{array}
$$

$$
\begin{array}{lll}
w2(\mathit{self}) & \triangleq & \land\, \mathit{pc}[\mathit{self}] = \text{“w2”} \\
 & & \land\, \lor\, \mathit{num}[\mathit{nxt}[\mathit{self}]] = 0 \\
 & & \quad \lor\, \langle \mathit{num}[\mathit{self}], \mathit{self} \rangle \prec \langle \mathit{num}[\mathit{nxt}[\mathit{self}]], \mathit{nxt}[\mathit{self}] \rangle \\
 & & \land\, \mathit{unchecked}' = [\mathit{unchecked} \text{ EXCEPT } ![\mathit{self}] = \mathit{unchecked}[\mathit{self}] \setminus \{\mathit{nxt}[\mathit{self}]\}] \\
 & & \land\, \mathit{pc}' = [\mathit{pc} \text{ EXCEPT } ![\mathit{self}] = \text{“w1”}] \\
 & & \land\, \text{UNCHANGED } \langle \mathit{num}, \mathit{flag}, \mathit{max}, \mathit{nxt} \rangle
\end{array}
$$

$$
\begin{array}{lll}
\mathit{cs}(\mathit{self}) & \triangleq & \land\, \mathit{pc}[\mathit{self}] = \text{“cs”} \\
 & & \land\, \text{TRUE} \\
 & & \land\, \mathit{pc}' = [\mathit{pc} \text{ EXCEPT } ![\mathit{self}] = \text{“exit”}] \\
 & & \land\, \text{UNCHANGED } \langle \mathit{num}, \mathit{flag}, \mathit{unchecked}, \mathit{max}, \mathit{nxt} \rangle
\end{array}
$$

$$
\begin{array}{lll}
\mathit{exit}(\mathit{self}) & \triangleq & \land\, \mathit{pc}[\mathit{self}] = \text{“exit”} \\
 & & \land\, \lor\, \land\, \exists\, k \in \mathit{Nat} : \\
 & & \quad\quad\quad\quad \mathit{num}' = [\mathit{num} \text{ EXCEPT } ![\mathit{self}] = k] \\
 & & \quad\;\;\, \land\, \mathit{pc}' = [\mathit{pc} \text{ EXCEPT } ![\mathit{self}] = \text{“exit”}] \\
 & & \quad \lor\, \land\, \mathit{num}' = [\mathit{num} \text{ EXCEPT } ![\mathit{self}] = 0] \\
 & & \quad\;\;\, \land\, \mathit{pc}' = [\mathit{pc} \text{ EXCEPT } ![\mathit{self}] = \text{“ncs”}] \\
 & & \land\, \text{UNCHANGED } \langle \mathit{flag}, \mathit{unchecked}, \mathit{max}, \mathit{nxt} \rangle
\end{array}
$$

$$
\begin{array}{lll}
p(\mathit{self}) & \triangleq & \mathit{ncs}(\mathit{self}) \lor e1(\mathit{self}) \lor e2(\mathit{self}) \lor e3(\mathit{self}) \lor e4(\mathit{self}) \\
 & & \quad \lor\, w1(\mathit{self}) \lor w2(\mathit{self}) \lor \mathit{cs}(\mathit{self}) \lor \mathit{exit}(\mathit{self})
\end{array}
$$

$$
\mathit{Next} \triangleq (\exists\, \mathit{self} \in \mathit{Procs} : p(\mathit{self}))
$$

$$
\mathit{Spec} \triangleq \land\, \mathit{Init} \land \Box[\mathit{Next}]_{\mathit{vars}}
$$

$$\land \forall self \in Procs : \mathrm{WF}_{vars}((pc[self] \neq \text{"ncs"}) \land p(self))$$
$$\land \forall self \in Procs : \mathrm{WF}_{vars}(\land e1(self) \lor e3(self) \lor e4(self) \lor exit(self) \\ \land (pc'[self] \neq pc[self]))$$

END TRANSLATION (this ends the translation of the *PlusCal* code)

MutualExclusion asserts that two distinct processes are in their critical sections.

$$MutualExclusion \triangleq \forall i, j \in Procs : (i \neq j) \Rightarrow \neg \land pc[i] = \text{"cs"} \\ \land pc[j] = \text{"cs"}$$

The Inductive Invariant

TypeOK is the type-correctness invariant.

$$\begin{aligned} TypeOK \triangleq\ & \land num \in [Procs \to Nat] \\ & \land flag \in [Procs \to \text{BOOLEAN}] \\ & \land unchecked \in [Procs \to \text{SUBSET } Procs] \\ & \land max \in [Procs \to Nat] \\ & \land nxt \in [Procs \to Procs] \\ & \land pc \in [Procs \to \{\text{"ncs"}, \text{"e1"}, \text{"e2"}, \text{"e3"}, \\ & \qquad \text{"e4"}, \text{"w1"}, \text{"w2"}, \text{"cs"}, \text{"exit"}\}] \end{aligned}$$

Before(i, j) is a condition that implies that $num[i] > 0$ and, if j is trying to enter its critical section and i does not change $num[i]$, then j either has or will choose a value of $num[j]$ for which

$\langle num[i], i\rangle \prec \langle num[j], j\rangle$

is true.

$$\begin{aligned} Before(i, j) \triangleq\ & \land num[i] > 0 \\ & \land \lor pc[j] \in \{\text{"ncs"}, \text{"e1"}, \text{"exit"}\} \\ & \quad \lor \land pc[j] = \text{"e2"} \\ & \qquad \land \lor i \in unchecked[j] \\ & \qquad \quad \lor max[j] \geq num[i] \\ & \quad \lor \land pc[j] = \text{"e3"} \\ & \qquad \land max[j] \geq num[i] \\ & \quad \lor \land pc[j] \in \{\text{"e4"}, \text{"w1"}, \text{"w2"}\} \\ & \qquad \land \langle num[i], i\rangle \prec \langle num[j], j\rangle \\ & \qquad \land (pc[j] \in \{\text{"w1"}, \text{"w2"}\}) \Rightarrow (i \in unchecked[j]) \end{aligned}$$

Inv is the complete inductive invariant.

$$\begin{aligned} Inv \triangleq\ & \land TypeOK \\ & \land \forall i \in Procs : \end{aligned}$$

$\land$ This conjunct is not needed for mutual exclusion, but it is needed to prove liveness.

$$\begin{aligned} & (pc[i] \in \{\text{"ncs"}, \text{"e1"}, \text{"e2"}\}) \Rightarrow (num[i] = 0) \\ & \land (pc[i] \in \{\text{"e4"}, \text{"w1"}, \text{"w2"}, \text{"cs"}\}) \Rightarrow (num[i] \neq 0) \\ & \land (pc[i] \in \{\text{"e2"}, \text{"e3"}\}) \Rightarrow flag[i] \end{aligned}$$

$\land$ This conjunct is not needed to prove mutual exclusion. It's needed to prove liveness, but it could be removed if the $\prec$ in the wait condition were changed to $\preceq$.

$$
\begin{array}{l}
\quad (pc[i] = \text{"w2"}) \Rightarrow (nxt[i] \neq i) \\
\land \; pc[i] \in \{\text{"e2"}, \text{"w1"}, \text{"w2"}\} \Rightarrow i \notin unchecked[i] \\
\land (pc[i] \in \{\text{"w1"}, \text{"w2"}\}) \Rightarrow \\
\qquad \forall j \in (Procs \setminus unchecked[i]) \setminus \{i\} : Before(i, j) \\
\land \; \land (pc[i] = \text{"w2"}) \\
\quad \land \lor (pc[nxt[i]] = \text{"e2"}) \land (i \notin unchecked[nxt[i]]) \\
\qquad \lor pc[nxt[i]] = \text{"e3"} \\
\quad \Rightarrow max[nxt[i]] \geq num[i] \\
\land (pc[i] = \text{"cs"}) \Rightarrow \forall j \in Procs \setminus \{i\} : Before(i, j)
\end{array}
$$

Proof of Mutual Exclusion

This is a standard invariance proof, where $\langle 1 \rangle 2$ asserts that any step of the algorithm (including a stuttering step) starting in a state in which *Inv* is true leaves *Inv* true. Step $\langle 1 \rangle 4$ follows easily from $\langle 1 \rangle 1 - \langle 1 \rangle 3$ by simple temporal reasoning, but *TLAPS* does not yet do any temporal reasoning.

$$
\begin{array}{l}
\textsc{theorem } Spec \Rightarrow \Box MutualExclusion \\
\langle 1 \rangle \textsc{ use } N \in Nat \textsc{defs } Procs, Inv, TypeOK, Before, \prec, ProcSet \\
\langle 1 \rangle 1.\ Init \Rightarrow Inv \\
\quad \textsc{by } SMT \textsc{ def } Init \\
\langle 1 \rangle 2.\ Inv \land [Next]_{vars} \Rightarrow Inv' \\
\quad \textsc{by } Z3 \textsc{ def } Next, ncs, p, e1, e2, e3, e4, w1, w2, cs, exit, vars \\
\langle 1 \rangle 3.\ Inv \Rightarrow MutualExclusion \\
\quad \textsc{by } SMT \textsc{ def } MutualExclusion \\
\langle 1 \rangle \textsc{ hide def } Inv \\
\langle 1 \rangle 4.\ \textsc{qed} \\
\quad \textsc{by } \langle 1 \rangle 1, \langle 1 \rangle 2, \langle 1 \rangle 3, PTL \textsc{ def } Spec
\end{array}
$$

$$
\begin{array}{l}
Trying(i) \triangleq pc[i] = \text{"e1"} \\
InCS(i) \triangleq pc[i] = \text{"cs"} \\
DeadlockFree \triangleq (\exists i \in Procs : Trying(i)) \leadsto (\exists i \in Procs : InCS(i)) \\
StarvationFree \triangleq \forall i \in Procs : Trying(i) \leadsto InCS(i)
\end{array}
$$

* Modification History
* Last modified *Tue Dec* 18 13:48:46 *PST* 2018 by *lamport*
* Created *Thu Nov* 21 15:54:32 *PST* 2013 by *lamport*