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 l 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
```

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 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.

$$a \prec b \triangleq \lor\, a[1] < b[1] \;\; \lor\, (a[1] = b[1]) \land (a[2] < b[2])$$

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

--algorithm *Bakery*

{ variable $num = [i \in Procs \mapsto 0]$, $flag = [i \in Procs \mapsto \text{FALSE}]$ **;**
  fair process ($p \in Procs$ **)**
    variables $unchecked = \{\}$, $max = 0$, $nxt = 1$ **;**
    { ncs:- **while (** TRUE **)**
          { $e1$: **either { ** $flag[self] := \neg flag[self]$ **;**
                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

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

$ProcSet \triangleq (Procs)$

$Init \triangleq$ Global variables
$\land num = [i \in Procs \mapsto 0]$
$\land flag = [i \in Procs \mapsto \text{FALSE}]$

Process p

$\land\ unchecked = [self \in Procs \mapsto \{\}]$
$\land\ max = [self \in Procs \mapsto 0]$
$\land\ nxt\ = [self \in Procs \mapsto 1]$
$\land\ pc = [self \in ProcSet \mapsto \text{“ncs”}]$

$ncs(self) \triangleq$
$\land\ pc[self] = \text{“ncs”}$
$\land\ pc' = [pc \text{ EXCEPT } ![self] = \text{“e1”}]$
$\land$ UNCHANGED $\langle num, flag, unchecked, max, nxt\rangle$

$e1(self) \triangleq$
$\land\ pc[self] = \text{“e1”}$
$\land\ \lor\ \land\ flag' = [flag \text{ EXCEPT } ![self] = \neg flag[self]]$
$\land\ pc' = [pc \text{ EXCEPT } ![self] = \text{“e1”}]$
$\land$ UNCHANGED $\langle unchecked, max\rangle$
$\lor\ \land\ flag' = [flag \text{ EXCEPT } ![self] = \text{TRUE}]$
$\land\ unchecked' = [unchecked \text{ EXCEPT } ![self] = Procs \setminus \{self\}]$
$\land\ max' = [max \text{ EXCEPT } ![self] = 0]$
$\land\ pc' = [pc \text{ EXCEPT } ![self] = \text{“e2”}]$
$\land$ UNCHANGED $\langle num, nxt\rangle$

$e2(self) \triangleq$
$\land\ pc[self] = \text{“e2”}$
$\land$ IF $unchecked[self] \neq \{\}$
THEN $\land\ \exists\, i \in unchecked[self] :$
$\land\ unchecked' = [unchecked \text{ EXCEPT } ![self] = unchecked[self] \setminus \{i\}]$
$\land$ IF $num[i] > max[self]$
THEN $\land\ max' = [max \text{ EXCEPT } ![self] = num[i]]$
ELSE $\land$ TRUE
$\land\ max' = max$
$\land\ pc' = [pc \text{ EXCEPT } ![self] = \text{“e2”}]$
ELSE $\land\ pc' = [pc \text{ EXCEPT } ![self] = \text{“e3”}]$
$\land$ UNCHANGED $\langle unchecked, max\rangle$
$\land$ UNCHANGED $\langle num, flag, nxt\rangle$

$e3(self) \triangleq$
$\land\ pc[self] = \text{“e3”}$
$\land\ \lor\ \land\ \exists\, k \in Nat :$
$num' = [num \text{ EXCEPT } ![self] = k]$
$\land\ pc' = [pc \text{ EXCEPT } ![self] = \text{“e3”}]$
$\lor\ \land\ \exists\, i \in \{j \in Nat : j > max[self]\} :$
$num' = [num \text{ EXCEPT } ![self] = i]$
$\land\ pc' = [pc \text{ EXCEPT } ![self] = \text{“e4”}]$
$\land$ UNCHANGED $\langle flag, unchecked, max, nxt\rangle$

$e4(self) \triangleq$
$\land\ pc[self] = \text{“e4”}$
$\land\ \lor\ \land\ flag' = [flag \text{ EXCEPT } ![self] = \neg flag[self]]$
$\land\ pc' = [pc \text{ EXCEPT } ![self] = \text{“e4”}]$
$\land$ UNCHANGED $unchecked$

$$
\begin{array}{l}
\quad \lor \land flag' = [flag \text{ EXCEPT } ![self] = \text{FALSE}] \\
\qquad \land unchecked' = [unchecked \text{ EXCEPT } ![self] = Procs \setminus \{self\}] \\
\qquad \land pc' = [pc \text{ EXCEPT } ![self] = \text{“w1”}] \\
\land \text{UNCHANGED } \langle num,\ max,\ nxt \rangle
\end{array}
$$

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

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

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

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

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

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

$$
\begin{array}{ll}
Spec \triangleq & \land Init \land \Box[Next]_{vars} \\
& \land \forall\, self \in Procs : \text{WF}_{vars}((pc[self] \neq \text{“ncs”}) \land p(self)) \\
& \land \forall\, self \in Procs : \text{WF}_{vars}(\ \land e1(self) \lor \ \ e3(self) \lor e4(self) \lor exit(self) \\
& \qquad\qquad\qquad\qquad\qquad\qquad \land (pc'[self] \neq pc[self]))
\end{array}
$$

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.

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

Inv is the complete inductive invariant.

$$\begin{aligned} Inv \triangleq\ & \land TypeOK \\ & \land \forall\, i \in Procs : \\ & \qquad \land (pc[i] \in \{\text{“ncs”}, \text{“e1”}, \text{“e2”}\}) \Rightarrow (num[i] = 0) \\ & \qquad \land (pc[i] \in \{\text{“e4”}, \text{“w1”}, \text{“w2”}, \text{“cs”}\}) \Rightarrow (num[i] \neq 0) \\ & \qquad \land (pc[i] \in \{\text{“e2”}, \text{“e3”}\}) \Rightarrow flag[i] \\ & \qquad \land (pc[i] = \text{“w2”}) \Rightarrow (nxt[i] \neq i) \\ & \qquad \land pc[i] \in \{\text{“e2”}, \text{“w1”}, \text{“w2”}\} \Rightarrow i \notin unchecked[i] \\ & \qquad \land (pc[i] \in \{\text{“w1”}, \text{“w2”}\}) \Rightarrow \\ & \qquad\quad \forall\, j \in (Procs \setminus unchecked[i]) \setminus \{i\} : Before(i, j) \\ & \qquad \land \land (pc[i] = \text{“w2”}) \\ & \qquad\quad \land \lor (pc[nxt[i]] = \text{“e2”}) \land (i \notin unchecked[nxt[i]]) \\ & \qquad\quad\ \lor pc[nxt[i]] = \text{“e3”} \\ & \qquad\quad \Rightarrow max[nxt[i]] \geq num[i] \\ & \qquad \land (pc[i] = \text{“cs”}) \Rightarrow \forall\, j \in Procs \setminus \{i\} : Before(i, j) \end{aligned}$$

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.

THEOREM $Spec \Rightarrow \Box MutualExclusion$
$\langle 1\rangle$ USE $N \in Nat$DEFS $Procs$, Inv, $TypeOK$, $Before$, $\prec$, $ProcSet$
$\langle 1\rangle 1$. $Init \Rightarrow Inv$
BY SMT DEF $Init$
$\langle 1\rangle 2$. $Inv \land [Next]_{vars} \Rightarrow Inv'$
BY $Z3$ DEF $Next$, ncs, p, $e1$, $e2$, $e3$, $e4$, $w1$, $w2$, cs, $exit$, $vars$
$\langle 1\rangle 3$. $Inv \Rightarrow MutualExclusion$
BY SMT DEF $MutualExclusion$
$\langle 1\rangle 4$. QED
BY $\langle 1\rangle 1$, $\langle 1\rangle 2$, $\langle 1\rangle 3$, PTL DEF $Spec$

$Trying(i) \triangleq pc[i] = \text{“e1”}$
$InCS(i) \triangleq pc[i] = \text{“cs”}$
$DeadlockFree \triangleq (\exists\, i \in Procs : Trying(i)) \leadsto (\exists\, i \in Procs : InCS(i))$
$StarvationFree \triangleq \forall\, i \in Procs : Trying(i) \leadsto InCS(i)$

* Modification History
* Last modified *Tue Jul* 21 17:55:24 *PDT* 2015 by *lamport*
* Created *Thu Nov* 21 15:54:32 *PST* 2013 by *lamport*