MODULE *Foo*

EXTENDS *Naturals*, *TLAPS*, *TLC*

CONSTANT N

ASSUME $NNat \triangleq N \in Nat \setminus \{0\}$

We'll write the algorithm in *PlusCal*, a powerful pseudocode-like imperative language, that compiles to TLA+. The translation (below), makes the semantics of this language crystal clear.

****

--algorithm *Foo* **{**

  variables $x = [i \in 0 \,..\, N-1 \mapsto 0]$,

    $y = [i \in 0 \,..\, N-1 \mapsto 0]$ **;**

  process ($Proc \in 0 \,..\, N-1$ **) {**

   $p1$: $x[self] := 1$ **;**

   $p2$: $y[self] := x[(self - 1)\%N]$ **;**

  }

}

****

BEGIN TRANSLATION

VARIABLES x, y, pc

$vars \triangleq \langle x, y, pc \rangle$

$ProcSet \triangleq (0 \,..\, N-1)$

$Init \triangleq$ Global variables

$\land x = [i \in 0 \,..\, N-1 \mapsto 0]$

$\land y = [i \in 0 \,..\, N-1 \mapsto 0]$

$\land pc = [self \in ProcSet \mapsto \text{“p1”}]$

$p1(self) \triangleq \land pc[self] = \text{“p1”}$

$\land x' = [x \text{ EXCEPT } ![self] = 1]$

$\land pc' = [pc \text{ EXCEPT } ![self] = \text{“p2”}]$

$\land y' = y$

$p2(self) \triangleq \land pc[self] = \text{“p2”}$

$\land y' = [y \text{ EXCEPT } ![self] = x[(self - 1)\%N]]$

$\land pc' = [pc \text{ EXCEPT } ![self] = \text{“Done”}]$

$\land x' = x$

$Proc(self) \triangleq p1(self) \lor p2(self)$

$Next \triangleq (\exists\, self \in 0 \,..\, N-1 : Proc(self))$

$\lor$ Disjunct to prevent deadlock on termination

$((\forall\, self \in ProcSet : pc[self] = \text{“Done”}) \land \text{UNCHANGED } vars)$

$Spec \triangleq Init \wedge \Box[Next]_{vars}$

$Termination \triangleq \Diamond(\forall self \in ProcSet : pc[self] = \text{"Done"})$

END TRANSLATION

That last definition, *Termination*, is automatically generated by the *PlusCal* compiler. It states the property that all processes eventually terminate. We won't use it.

A few helper definitions:

$Prev(p) \triangleq (p - 1)\%N$

LEMMA $PrevInSet \triangleq \forall i \in ProcSet : Prev(i) \in ProcSet$
BY $NNat$ DEF $ProcSet$, $Prev$

$Done(p) \triangleq pc[p] = \text{"Done"}$

$q \sqsubseteq t \triangleq 1$

$AllDone \triangleq \forall p \in ProcSet : Done(p)$

A "type" invariant

$$
\begin{aligned}
TypeOK \triangleq\ & \wedge pc \in [ProcSet \rightarrow \{\text{"p1"}, \text{"p2"}, \text{"Done"}\}] \\
& \wedge x \ \in [ProcSet \rightarrow \{0, 1\}] \\
& \wedge y \ \in [ProcSet \rightarrow \{0, 1\}]
\end{aligned}
$$

The algorithm's property:

When all processes have terminated, at least one of the y's is 1

$PartialCorrectness \triangleq AllDone \Rightarrow \exists p \in ProcSet : y[p] = 1$

We can use *TLC*, a TLA+ model checker to test that *PartialCorrectness* is indeed an invariant. . . . yep.

This is a sufficient inductive invariant:

$$
\begin{aligned}
Inv \triangleq\ & \wedge TypeOK \\
& \wedge PartialCorrectness \\
& \wedge \forall p \in ProcSet : pc[p] \neq \text{"p1"} \Rightarrow x[p] = 1
\end{aligned}
$$

Inv is an inductive invariant of *Next* iff it is an ordinary invariant of the specification *ISpec*:

$ISpec \triangleq Inv \wedge \Box[Next]_{vars}$

It's easier to prove something if it's true, so we use *TLC* to check that *Inv* is an inductive invariant. . . . yep.

The main proof.

TLA+ proofs use a declarative language based on *Lamport*'s structured "modern" proofs style. See: http://*research.microsoft.com*/en-us/um/people/lamport/pubs/*proof.pdf*

When writing them in the TLA+ *IDE*, the proof levels are collapsible, and all the names and labels hyperlinked.

THEOREM $Spec \Rightarrow \Box PartialCorrectness$

Make some defs and lemmas automatically available to the proof backends

$\langle 1 \rangle$ USE $PrevInSet$ DEF $AllDone$, $Done$, $Prev$, $ProcSet$

First we prove that the invariant implies the property; it ' s trivial in our case

$\langle 1 \rangle 1.$ $Inv \Rightarrow PartialCorrectness$BY DEF Inv

Now we prove that Inv is indeed an inductive invariant

$\langle 1 \rangle 2.$ $Init \Rightarrow Inv$ It holds in the initial state

$\langle 2 \rangle$ HAVE $Init$

$\langle 2 \rangle 1.$ $0 \in ProcSet$BY $NNat$

$\langle 2 \rangle 2.$ $\neg AllDone$BY $\langle 2 \rangle 1$ DEF $Init$

$\langle 2 \rangle 3.$ QED BY $\langle 2 \rangle 2$ DEF $Init$, Inv, $TypeOK$, $PartialCorrectness$

$\langle 1 \rangle 3.$ $Inv \land [Next]_{vars} \Rightarrow Inv'$ It is inductive, so it holds in all states $\Rightarrow$ invariant

$\langle 2 \rangle$ SUFFICES ASSUME Inv, $[Next]_{vars}$PROVE Inv'OBVIOUS

Case 1: a process makes a $p1$ step

$\langle 2 \rangle 1.$ ASSUME NEW $self \in ProcSet$, $p1(self)$PROVE Inv'

$\langle 3 \rangle$ SUFFICES ASSUME $p1(self)$PROVE Inv'BY $\langle 2 \rangle 1$

$\langle 3 \rangle 1.$ $TypeOK'$BY DEF $p1$, Inv, $TypeOK$ We need to prove type preservation, but that's easy

$\langle 3 \rangle 2.$ $\forall p \in ProcSet \setminus \{self\} : x[p] = x'[p] \land y[p] = y'[p] \land pc[p] = pc'[p]$
BY DEF $p1$, Inv, $TypeOK$

$\langle 3 \rangle 3.$ $Done(self)' \equiv Done(self)$BY $\langle 3 \rangle 1$ DEF $p1$, $TypeOK$

No new processes are done

$\langle 3 \rangle 4.$ $AllDone' \equiv AllDone$BY $\langle 3 \rangle 2$, $\langle 3 \rangle 3$ DEF $TypeOK$ unnecessary step

so $PartialCorrectness$ is preserved

$\langle 3 \rangle 5.$ $PartialCorrectness'$BY $\langle 3 \rangle 2$, $\langle 3 \rangle 3$ DEF $p1$, $PartialCorrectness$, Inv

$\langle 3 \rangle 6.$ $pc[self]' \neq \text{“p1”} \land x[self]' = 1$BY $\langle 3 \rangle 1$ DEF $p1$, $TypeOK$

$\langle 3 \rangle 7.$ QED BY $\langle 3 \rangle 1$, $\langle 3 \rangle 2$, $\langle 3 \rangle 5$, $\langle 3 \rangle 6$ DEF Inv, $TypeOK$

Case 2: a process makes a $p2$ step

$\langle 2 \rangle 2.$ ASSUME NEW $self \in ProcSet$, $p2(self)$PROVE Inv'

$\langle 3 \rangle$ SUFFICES ASSUME $p2(self)$PROVE Inv'BY $\langle 2 \rangle 2$

$\langle 3 \rangle 1.$ $TypeOK'$BY DEF $p2$, Inv, $TypeOK$ We need to prove type preservation, but that's easy

$\langle 3 \rangle 2.$ $\forall p \in ProcSet \setminus \{self\} : x[p] = x'[p] \land y[p] = y'[p] \land pc[p] = pc'[p]$
BY DEF $p2$, Inv, $TypeOK$

$\langle 3 \rangle 3.$ $Done(self)'$BY $\langle 3 \rangle 1$ DEF $TypeOK$, $Done$, $p2$

$\langle 3 \rangle 4.$ $x'[self] = 1$BY $\langle 3 \rangle 1$ DEF Inv, $p2$

$\langle 3 \rangle 5.$ $(\forall p \in ProcSet : pc[p] \neq \text{“p1”} \Rightarrow x[p] = 1)'$

$\langle 4 \rangle 1.$ $\forall p \in ProcSet \setminus \{self\} : (pc[p] \neq \text{“p1”} \Rightarrow x[p] = 1)'$
BY $\langle 3 \rangle 1$, $\langle 3 \rangle 2$ DEF Inv

$\langle 4 \rangle 2.$ $(pc[self] \neq \text{“p1”} \Rightarrow x[self] = 1)'$
BY $\langle 3 \rangle 1$ DEF $p2$, Inv, $TypeOK$

$\langle 4 \rangle 3.$ QED BY $\langle 4 \rangle 1$, $\langle 4 \rangle 2$

$\langle 3 \rangle 6.$ $y[self]' = x[Prev(self)]$BY $\langle 3 \rangle 1$ DEF $p2$, $TypeOK$

$\langle 3 \rangle 7.$ $x[Prev(self)] \neq 1 \Rightarrow \neg Done(Prev(self))$BY DEF Inv

$\langle 3 \rangle 8.$ $PartialCorrectness'$

Either I assigned 1 to y

$\langle 4\rangle 2.$CASE $y[self]' = 1$BY $\langle 4\rangle 2$ DEF Inv, $PartialCorrectness$

.. or I assigned 0 and this means $Prev(self)$ is not done

$\langle 4\rangle 1.$CASE $y[self]' = 0$

$\langle 5\rangle 1.$ $\neg Done(Prev(self))$
BY $\langle 4\rangle 1$, $\langle 3\rangle 6$, $\langle 3\rangle 7$ DEF $p2$

$\langle 5\rangle 2.$ $x[self] = 1$BY DEF Inv, $p2$

$\langle 5\rangle 3.$ $Prev(self) = self \Rightarrow y[self]' = 1$
BY Inv DEF $p2$, Inv, $TypeOK$

$\langle 5\rangle 4.$ $Prev(self) \neq self$BY $\langle 5\rangle 3$, $\langle 4\rangle 1$

$\langle 5\rangle 5.$ $Prev(self) \in ProcSet \setminus \{self\}$BY $\langle 5\rangle 4$, $PrevInSet$

$\langle 5\rangle 6.$ $\neg AllDone'$BY $PrevInSet$, $\langle 5\rangle 1$, $\langle 5\rangle 5$, $\langle 3\rangle 2$

$\langle 5\rangle 7.$ QED BY $\langle 5\rangle 6$ DEF Inv, $PartialCorrectness$

$\langle 4\rangle 3.$ QED BY $\langle 4\rangle 1$, $\langle 4\rangle 2$, $\langle 3\rangle 1$ DEF $TypeOK$

$\langle 3\rangle 9.$ QED BY $\langle 3\rangle 1$, $\langle 3\rangle 5$, $\langle 3\rangle 8$ DEF Inv

Trivial cases:

$\langle 2\rangle 3.$CASE $(\forall\, self \in ProcSet : pc[self] = \text{“Done”}) \land$ UNCHANGED $vars$
BY $\langle 2\rangle 3$ DEF Inv, $TypeOK$, $vars$, $PartialCorrectness$

$\langle 2\rangle 4.$CASE UNCHANGED $vars$BY $\langle 2\rangle 4$ DEF Inv, $TypeOK$, $vars$, $PartialCorrectness$

$\langle 2\rangle 5.$ QED BY $\langle 2\rangle 1$, $\langle 2\rangle 2$, $\langle 2\rangle 3$, $\langle 2\rangle 4$ DEF $Next$, $Proc$

$\langle 1\rangle 4.$ QED BY $\langle 1\rangle 2$, $\langle 1\rangle 3$, $\langle 1\rangle 1$, PTL DEF $Spec$