

— MODULE *Fischer* —

This module contains defines the formula *Fischer*, which describes Fischers algorithm of Sections 5.2.1 – 5.2.3 of “A Science of Computer Programming” by *Leslie Lamport*. *TLC* can be used to check that this algorithm satisfies mutual exclusion, as well as the invariant in formula (5.2) that is needed to reason about liveness. However, liveness is not considered.

Since *TLC* cannot handle real numbers, this module defines a discrete-time version of *Fischer*’s algorithm in which the set *Times* of all possible times is a set of natural numbers.

EXTENDS *Integers*

CONSTANT *Procs*, *delta*, *epsilon*, *Times*

We assume that *Times* is a subset of the set *Nat* of natural numbers, and *delta* and *epsilon* are natural numbers. And we let all behaviors begin at time 0. The discrete-time version of *Fischer*’s algorithm is obtained by letting *Time* equal *Nat*. Letting *Time* be a finite interval of natural numbers produces a finite-state model on which *TLC* can check invariants.

ASSUME  $\wedge \{delta, epsilon\} \subseteq Nat$   
 $\wedge Times \subseteq Nat$

We define *none* to be some unspecified value that is not in *Procs*.

$none \triangleq \text{CHOOSE } n : n \notin Procs$

VARIABLE *x*, *pc*, *rt*, *now*

$v \triangleq \langle x, pc, rt, now \rangle$

$Init \triangleq \wedge x = none$   
 $\wedge pc = [p \in Procs \mapsto \text{“wait”}]$   
 $\wedge rt = [p \in Procs \mapsto 0]$   
 $\wedge now = 0$

$Ncs(p) \triangleq \wedge pc[p] = \text{“ncs”}$   
 $\wedge pc' = [pc \text{ EXCEPT } ![p] = \text{“wait”}]$   
 $\wedge \text{UNCHANGED } \langle x, rt, now \rangle$

$Wait(p) \triangleq \wedge pc[p] = \text{“wait”}$   
 $\wedge x = none$   
 $\wedge rt' = [rt \text{ EXCEPT } ![p] = now]$   
 $\wedge pc' = [pc \text{ EXCEPT } ![p] = \text{“w1”}]$   
 $\wedge \text{UNCHANGED } \langle x, now \rangle$

$W1(p) \triangleq \wedge pc[p] = \text{“w1”}$   
 $\wedge x' = p$   
 $\wedge rt' = [rt \text{ EXCEPT } ![p] = now]$   
 $\wedge pc' = [pc \text{ EXCEPT } ![p] = \text{“w2”}]$   
 $\wedge now' = now$

$W2(p) \triangleq \wedge pc[p] = \text{“w2”}$   
 $\wedge now - rt[p] \geq epsilon$   
 $\wedge pc' = [pc \text{ EXCEPT } ![p] = \text{IF } x = p \text{ THEN “cs” ELSE “wait”}]$

$$\begin{aligned}
& \wedge \text{UNCHANGED } \langle x, rt, now \rangle \\
Cs(p) & \triangleq \wedge pc[p] = \text{"cs"} \\
& \wedge pc' = [pc \text{ EXCEPT } ![p] = \text{"exit"}] \\
& \wedge \text{UNCHANGED } \langle x, rt, now \rangle \\
Exit(p) & \triangleq \wedge pc[p] = \text{"exit"} \\
& \wedge x' = none \\
& \wedge pc' = [pc \text{ EXCEPT } ![p] = \text{"ncs"}] \\
& \wedge \text{UNCHANGED } \langle rt, now \rangle \\
Time & \triangleq \wedge now' \in \{t \in Times : t > now\} \\
& \wedge \forall p \in Procs : (pc[p] = \text{"w1"}) \Rightarrow (now' \leq rt[p] + delta) \\
& \wedge \text{UNCHANGED } \langle x, pc, rt \rangle \\
PNext(p) & \triangleq Ncs(p) \vee Wait(p) \vee W1(p) \vee W2(p) \vee Cs(p) \vee Exit(p) \\
Next & \triangleq \vee \exists p \in Procs : PNext(p) \\
& \vee Time \\
Fischer & \triangleq Init \wedge \Box[Next]_v
\end{aligned}$$

---

We define the formula *Inv* consisting of the conjunction of a type-correctness invariant *TypeOK* and the formula at the beginning of Section 5.2.2 asserting three properties. *TLC* can check that *Inv* is an inductive invariant for small models (a small set *Time* and few processes).

$$\begin{aligned}
TypeOK & \triangleq \wedge x \in Procs \cup \{none\} \\
& \wedge pc \in [Procs \rightarrow \{\text{"ncs"}, \text{"wait"}, \text{"w1"}, \text{"w2"}, \text{"cs"}, \text{"exit"}\}] \\
& \wedge rt \in [Procs \rightarrow Times] \\
& \wedge now \in Times \\
Inv & \triangleq \wedge TypeOK \\
& \wedge \forall p \in Procs : \\
& \quad \wedge (pc[p] = \text{"w1"}) \Rightarrow (now \in rt[p] .. (rt[p] + delta)) \\
& \quad \wedge (pc[p] \in \{\text{"cs"}, \text{"exit"}\}) \Rightarrow \\
& \quad \quad (x = p) \wedge \forall j \in Procs : pc[j] \neq \text{"w1"} \\
& \quad \wedge (pc[p] = \text{"w2"}) \wedge (x = p) \Rightarrow \\
& \quad \quad \forall q \in Procs : (pc[q] = \text{"w1"}) \Rightarrow \\
& \quad \quad \quad (rt[q] + delta) < (rt[p] + epsilon)
\end{aligned}$$

We define the mutual exclusion invariant *Mutex*. *TLAPS* easily proves that *Inv* implies *Mutex*, so the invariance of *Inv* implies that *Fischer's* algorithm satisfies mutual exclusion.

$$Mutex \triangleq \forall p, q \in Procs : (p \neq q) \Rightarrow \{pc[p], pc[q]\} \neq \{\text{"cs"}\}$$

INSTANCE *TLAPS*  
THEOREM  $Inv \Rightarrow Mutex$   
BY DEF *Inv*, *TypeOK*, *Mutex*

---

Finally, we define  $LInv$  to be the invariant that is the conjunction of  $Inv$  with the invariant formula (5.2).  $TLC$  can check that it is an inductive invariant.

$$LInv \triangleq \wedge Inv \\ \wedge (x \neq none) \Rightarrow (pc[x] \in \{ \text{"w2"}, \text{"cs"}, \text{"exit"} \})$$

---

\ \* Modification History  
\ \* Last modified *Thu Oct 24 11:42:57 CEST 2024* by *lamport*  
\ \* Created *Wed Aug 03 18:14:42 PDT 2022* by *lamport*