MODULE *init_once*

EXTENDS *TLC*, *FiniteSets*, *Integers*

CONSTANTS *Processes*

--algorithm *init_once*
variables
lock = FALSE ;
is_init = FALSE ;

pids = {} ;

define
just_once $\triangleq$ $\Diamond\Box(Cardinality(pids) = 1) \land \Box(Cardinality(pids) \leq 1)$
end define

initializer
fair + **process** *pid* $\in$ *Processes*
begin
start_init_once:
while $\neg$*is_init* **do**
load_lock_relaxed:
if $\neg$*lock* **then**
compare_exchange:
if $\neg$*lock* **then**
lock := TRUE ;
pids := *pids* $\cup$ {*self*} ;

initialize:
skip ;

store_is_init:
is_init := TRUE ;
end if ;
end if ;
end while ;
end process ;

end algorithm ;
BEGIN TRANSLATION ($chksum(pcal) =$ "4542d7b0" $\land\ chksum(tla) =$ "4af7d3b")
VARIABLES *lock*, *is_init*, *pids*, *pc*

define statement
just_once $\triangleq$ $\Diamond\Box(Cardinality(pids) = 1) \land \Box(Cardinality(pids) \leq 1)$

vars $\triangleq$ $\langle$*lock*, *is_init*, *pids*, *pc*$\rangle$

ProcSet $\triangleq$ (*Processes*)

$Init \triangleq$ Global variables
$\land\ lock = \text{FALSE}$
$\land\ is\_init = \text{FALSE}$
$\land\ pids = \{\}$
$\land\ pc = [self \in ProcSet \mapsto \text{“start\_init\_once”}]$

$start\_init\_once(self) \triangleq \land\ pc[self] = \text{“start\_init\_once”}$
$\land\ \text{IF } \neg is\_init$
$\text{THEN } \land\ pc' = [pc \text{ EXCEPT } ![self] = \text{“load\_lock\_relaxed”}]$
$\text{ELSE } \land\ pc' = [pc \text{ EXCEPT } ![self] = \text{“Done”}]$
$\land\ \text{UNCHANGED } \langle lock, is\_init, pids \rangle$

$load\_lock\_relaxed(self) \triangleq \land\ pc[self] = \text{“load\_lock\_relaxed”}$
$\land\ \text{IF } \neg lock$
$\text{THEN } \land\ pc' = [pc \text{ EXCEPT } ![self] = \text{“compare\_exchange”}]$
$\text{ELSE } \land\ pc' = [pc \text{ EXCEPT } ![self] = \text{“start\_init\_once”}]$
$\land\ \text{UNCHANGED } \langle lock, is\_init, pids \rangle$

$compare\_exchange(self) \triangleq \land\ pc[self] = \text{“compare\_exchange”}$
$\land\ \text{IF } \neg lock$
$\text{THEN } \land\ lock' = \text{TRUE}$
$\land\ pids' = (pids \cup \{self\})$
$\land\ pc' = [pc \text{ EXCEPT } ![self] = \text{“initialize”}]$
$\text{ELSE } \land\ pc' = [pc \text{ EXCEPT } ![self] = \text{“start\_init\_once”}]$
$\land\ \text{UNCHANGED } \langle lock, pids \rangle$
$\land\ \text{UNCHANGED } is\_init$

$initialize(self) \triangleq \land\ pc[self] = \text{“initialize”}$
$\land\ \text{TRUE}$
$\land\ pc' = [pc \text{ EXCEPT } ![self] = \text{“store\_is\_init”}]$
$\land\ \text{UNCHANGED } \langle lock, is\_init, pids \rangle$

$store\_is\_init(self) \triangleq \land\ pc[self] = \text{“store\_is\_init”}$
$\land\ is\_init' = \text{TRUE}$
$\land\ pc' = [pc \text{ EXCEPT } ![self] = \text{“start\_init\_once”}]$
$\land\ \text{UNCHANGED } \langle lock, pids \rangle$

$pid(self) \triangleq start\_init\_once(self) \lor load\_lock\_relaxed(self)$
$\lor\ compare\_exchange(self) \lor initialize(self)$
$\lor\ store\_is\_init(self)$

Allow infinite stuttering to prevent deadlock on termination.
$Terminating \triangleq \land\ \forall\, self \in ProcSet : pc[self] = \text{“Done”}$
$\land\ \text{UNCHANGED } vars$

$Next \triangleq (\exists\, self \in Processes : pid(self))$
$\lor\ Terminating$

$$Spec \triangleq \land Init \land \Box[Next]_{vars} \\ \land \forall self \in Processes : \mathrm{SF}_{vars}(pid(self))$$

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

END TRANSLATION