
MODULE *ICenSeq1*

The first part of this module defines the TLA+ representation of the abstract program *ICenSeq1* of Figure 7.6 of the book “A Science of Computer Programs” by *Leslie Lamport*. The second part defines the program *ICenSeq1_p* obtained by adding a prophecy sequence variable *p* to *ICenSeq1* as described in Section 7.4.3.1, and it asserts that *ICenSeq1_p* implements *ICenSeq2* under a suitable refinement mapping. For that purpose, it instantiates module *ICenSeq2* that defines program *ICenSeq2*.

EXTENDS *Integers, Sequences*

CONSTANT *Art, NotArt*

VARIABLES *inp, aw, disp*
 $vars \triangleq \langle inp, aw, disp \rangle$

$TypeOKSeq \triangleq \wedge inp \in Art \cup \{NotArt\}$
 $\wedge disp \in Art \times \{0, 1\}$
 $\wedge aw \in Seq(Art)$

$InitSeq \triangleq \wedge inp = NotArt$
 $\wedge aw = \langle \rangle$
 $\wedge disp \in Art \times \{0, 1\}$

$InputSeq \triangleq \wedge inp = NotArt$
 $\wedge inp' \in Art$
 $\wedge aw' = Append(aw, inp')$
 $\wedge disp' = disp$

$AckSeq \triangleq \wedge inp \in Art$
 $\wedge inp' = NotArt$
 $\wedge UNCHANGED \langle aw, disp \rangle$

$DispOrNotSeq \triangleq \wedge aw \neq \langle \rangle$
 $\wedge \vee disp' = \langle aw[1], 1 - disp[2] \rangle$
 $\vee disp' = disp$
 $\wedge aw' = Tail(aw)$
 $\wedge UNCHANGED inp$

$NextSeq1 \triangleq InputSeq \vee AckSeq \vee DispOrNotSeq$

Although not done in the book, we define a fairness requirement for the *ICenSeq1* program. It is weak fairness of all the actions except the *InputSeq* action, since we don't want to require that the artist keep submitting works of art.

$Fairness1 \triangleq WF_{vars}(AckSeq \vee DispOrNotSeq)$

$ICenSeq1 \triangleq InitSeq \wedge \Box[NextSeq1]_{vars} \wedge Fairness1$

We now define the program *ICenSeq1_p* obtained by adding a prophecy sequence variable *p* to *ICenSeq1* as described in Section 7.4.3.2.

CONSTANTS Yes, No
 $Pi \triangleq \{Yes, No\}$
 ASSUME $Yes \neq No$
 $DorNSeq(i) \triangleq \wedge aw \neq \langle \rangle$
 $\wedge \vee (i = Yes) \wedge (disp' = \langle aw[1], 1 - disp[2] \rangle)$
 $\vee (i = No) \wedge (disp' = disp)$
 $\wedge aw' = Tail(aw)$
 $\wedge \text{UNCHANGED } inp$

VARIABLE p
 $vars_p \triangleq \langle inp, aw, disp, p \rangle$
 $TypeOKSeq_p \triangleq TypeOKSeq \wedge (p \in Seq(Pi))$
 $InitSeq_p \triangleq InitSeq \wedge (p = \langle \rangle)$
 $InputSeq_p \triangleq InputSeq \wedge \exists i \in Pi : p' = Append(p, i)$
 $AckSeq_p \triangleq AckSeq \wedge (p' = p)$
 $DispOrNotSeq_p \triangleq DorNSeq(p[1]) \wedge (p' = Tail(p))$
 $NextSeq1_p \triangleq InputSeq_p \vee AckSeq_p \vee DispOrNotSeq_p$
 $ICenSeq1_p \triangleq InitSeq_p \wedge \square [NextSeq1_p]_{vars_p} \wedge Fairness1$

We now define the mapping *OnlyYes* and state predicate *awBar* of Section 7.4.3.2 of the book.

RECURSIVE $OnlyYes(-, -)$
 $OnlyYes(ws, ys) \triangleq$
 $\quad \text{IF } ws = \langle \rangle \text{ THEN } \langle \rangle$
 $\quad \text{ELSE } (\text{IF } Head(ys) = Yes \text{ THEN } \langle Head(ws) \rangle$
 $\quad \quad \quad \text{ELSE } \langle \rangle$
 $\quad \quad \quad) \circ OnlyYes(Tail(ws), Tail(ys))$
 $awBar \triangleq OnlyYes(aw, p)$

We now assert the theorem (7.16).

$IC2 \triangleq \text{INSTANCE } ICenSeq2 \text{ WITH } aw \leftarrow awBar$
 THEOREM $ICenSeq1_p \Rightarrow IC2!ICenSeq2$

\ * Modification History
 \ * Last modified *Fri Oct 18 15:24:26 CEST 2024* by *lamport*
 \ * Created *Fri Oct 21 06:36:45 PDT 2022* by *lamport*