producerConsumer

Tight Rope v0.75

26th February 2017

1 ID Files

1.1 MissionIds

section *MissionIds* **parents** *scj_prelude*, *MissionId*

$$PCMissionMID : MissionID$$

$$distinct\langle nullMissionId, PCMissionMID\rangle$$

1.2 SchedulablesIds

section *SchedulableIds* **parents** *scj_prelude*, *SchedulableId*

$$
\begin{array}{l}
PCMissionSequencerSID : SchedulableID \\
ProducerSID : SchedulableID \\
ConsumerSID : SchedulableID \\
\hline
distinct\langle nullSequencerId, nullSchedulableId, PCMissionSequencerSID, \\
ProducerSID, ConsumerSID\rangle
\end{array}
$$

1.3 Non-Paradigm Objects

section *BufferApp* **parents** *scj_prelude*, *SchedulableId*, *SchedulableIds*, *SafeletChan*
, *BufferClass*, *MethodCallBindingChannels* , *ObjectChan*, *ObjectIds*, *ThreadIds*, *ObjectFWChan*, *ObjectIds*

$BufferID : NonParadigmID$

process $BufferApp \mathrel{\widehat=}$ **begin**

State

$this : \mathbf{ref}\ BufferClass$

state *State*

Init

$State'$

$this' = \mathbf{new}\ BufferClass()$

$$
\begin{array}{l}
writeSyncMeth \mathrel{\widehat=} \\
\left(\begin{array}{l}
writeCall\,.\,BufferID\,?\,caller\,?\,thread\,?\,update \longrightarrow \\
\left(\begin{array}{l}
startSyncMeth\,.\,BufferOID\,.\,thread \longrightarrow \\
lockAcquired\,.\,BufferOID\,.\,thread \longrightarrow \\
\left(\begin{array}{l}
\mu X \bullet \\
\left(\begin{array}{l}
\mathbf{var}\ loopVar : \mathbb{B} \bullet loopVar := (\neg\ this\,.\,bufferEmpty()); \\
\mathbf{if}\ (loopVar = \mathbf{True}) \longrightarrow \\
\qquad \left(\begin{array}{l}
\mathbf{Skip}; \\
waitCall\,.\,BufferOID\,.\,thread \longrightarrow \\
waitRet\,.\,BufferOID\,.\,thread \longrightarrow \\
\mathbf{Skip}
\end{array}\right) ;\ X \\
[\!]\ (loopVar = \mathbf{False}) \longrightarrow \mathbf{Skip} \\
\mathbf{fi}
\end{array}\right) \\
; \\
\mathbf{Skip}; \\
theBuffer := update; \\
notify\,.\,BufferOID\,!\,thread \longrightarrow \\
\mathbf{Skip}
\end{array}\right) ; \\
endSyncMeth\,.\,BufferOID\,.\,thread \longrightarrow \\
writeRet\,.\,BufferID\,.\,caller\,.\,thread \longrightarrow \\
\mathbf{Skip}
\end{array}\right)
\end{array}
$$

$$
\begin{array}{l}
readSyncMeth \mathrel{\widehat=} \mathbf{var}\ ret : \mathbb{Z} \bullet \\
\left(\begin{array}{l}
readCall\,.\,BufferID\,?\,caller\,?\,thread \longrightarrow \\
\left(\begin{array}{l}
startSyncMeth\,.\,BufferOID\,.\,thread \longrightarrow \\
lockAcquired\,.\,BufferOID\,.\,thread \longrightarrow \\
\left(\begin{array}{l}
\mu X \bullet \\
\left(\begin{array}{l}
\mathbf{var}\ loopVar : \mathbb{B} \bullet loopVar := \ this\,.\,bufferEmpty(); \\
\mathbf{if}\ (loopVar = \mathbf{True}) \longrightarrow \\
\qquad \left(\begin{array}{l} waitCall\,.\,BufferOID\,.\,thread \longrightarrow \\ waitRet\,.\,BufferOID\,.\,thread \longrightarrow \end{array}\right) ;\ X \\
\qquad \mathbf{Skip} \\
[\!]\ (loopVar = \mathbf{False}) \longrightarrow \mathbf{Skip} \\
\mathbf{fi}
\end{array}\right) \\
;\\
\mathbf{var}\ out : \mathbb{Z} \bullet out := this\,.\,theBuffer; \\
\mathbf{Skip}; \\
ABthis\,.\,theBuffer := 0; \\
notify\,.\,BufferOID\,!\,thread \longrightarrow \\
\mathbf{Skip}; \\
ret := out
\end{array}\right) \\
endSyncMeth\,.\,BufferOID\,.\,thread \longrightarrow \\
readRet\,.\,BufferID\,.\,caller\,.\,thread\,!\,ret \longrightarrow \\
\mathbf{Skip}
\end{array}\right) ;
\end{array}\right)
\end{array}
$$

$$
\begin{array}{l}
Methods \mathrel{\widehat=} \\
\left(\begin{array}{l}
GetSequencer \\
\Box \\
InitializeApplication \\
\Box \\
writeSyncMeth \\
\Box \\
readSyncMeth
\end{array}\right) ;\ Methods
\end{array}
$$

$$
\bullet\ (Init\ ;\ Methods) \triangle (end\_safelet\_app \longrightarrow \mathbf{Skip})
$$

end

section *BufferClass* **parents** *scj_prelude*, *SchedulableId*, *SchedulableIds*, *SafeletChan*
, *MethodCallBindingChannels*

class *BufferClass* $\widehat{=}$ **begin**

state *State*

$$theBuffer : \mathbb{Z}$$

state *State*

initial *Init*

$$State'$$

$$theBuffer' = 0$$

public *bufferEmpty* $\widehat{=}$

$$\left(\begin{array}{l}\textbf{Skip};\\ \textbf{if}\ (theBuffer = 0) \longrightarrow\\ \quad ret := \textbf{True}\\ [\!]\ \neg\ (theBuffer = 0) \longrightarrow\\ \quad ret := \textbf{False}\\ \textbf{fi}\end{array}\right)$$

- **Skip**

end

1.4 ThreadIds

section *ThreadIds* **parents** *scj_prelude*, *GlobalTypes*

$$
\begin{array}{|l}
SafeletTId : ThreadID \\
nullThreadId : ThreadID \\
ProducerTID : ThreadID \\
ConsumerTID : ThreadID \\
\hline
distinct\langle SafeletTId, nullThreadId, \\
ProducerTID, ConsumerTID\rangle
\end{array}
$$

1.5 ObjectIds

section *ObjectIds* **parents** *scj_prelude*, *GlobalTypes*

$$BufferOID : ObjectID$$

$$distinct\langle BufferOID\rangle$$

2 Network

2.1 Network Channel Sets

section *NetworkChannels* **parents** *scj_prelude*, *MissionId*, *MissionIds*, *SchedulableId*, *SchedulableIds*, *MissionChan*, *TopLevelMissionSequencerFWChan*, *FrameworkChan*, *SafeletChan*, *AperiodicEventHandlerChan*, *ManagedThreadChan*, *OneShotEventHandlerChan*, *PeriodicEventHandlerChan*, *MissionSequencerMethChan*

channelset *TerminateSync* ==
{| *schedulables_terminated*, *schedulables_stopped*, *get_activeSchedulables* |}

channelset *ControlTierSync* ==
{| *start_toplevel_sequencer*, *done_toplevel_sequencer*, *done_safeletFW* |}

channelset *TierSync* ==
{| *start_mission* . *PCMission*, *done_mission* . *PCMission*, *done_safeletFW*, *done_toplevel_sequencer* |}

channelset *MissionSync* ==
{| *done_safeletFW*, *done_toplevel_sequencer*, *register*, *signalTerminationCall*, *signalTerminationRet*, *activate_schedulables*, *done_schedulable*, *cleanupSchedulableCall*, *cleanupSchedulableRet* |}

channelset *SchedulablesSync* ==
{| *activate_schedulables*, *done_safeletFW*, *done_toplevel_sequencer* |}

channelset *ClusterSync* ==
{| *done_toplevel_sequencer*, *done_safeletFW* |}

channelset *SafeltAppSync* $\widehat{=}$
{| *getSequencerCall*, *getSequencerRet*, *initializeApplicationCall*, *initializeApplicationRet*, *end_safelet_app* |}

channelset *MissionSequencerAppSync* ==
{| *getNextMissionCall*, *getNextMissionRet*, *end_sequencer_app* |}

channelset *MissionAppSync* ==
{| *initializeCall*, *register*, *initializeRet*, *cleanupMissionCall*, *cleanupMissionRet* |}

channelset *AppSync* ==
$\bigcup\{$*SafeltAppSync*, *MissionSequencerAppSync*, *MissionAppSync*, *MTAppSync*, *OSEHSync*, *APEHSync*, *PEHSync*, {| *getSequencer*, *end_mission_app*, *end_managedThread_app*, *setCeilingPriority*, *requestTerminationCall*, *requestTerminationRet*, *terminationPendingCall*, *terminationPendingRet*, *handleAsyncEventCall*, *handleAsyncEventRet* |}$\}$

channelset *ThreadSync* ==
{| *raise_thread_priority*, *lower_thread_priority*, *isInterruptedCall*, *isInterruptedRet*, *get_priorityLevel* |}

channelset *LockingSync* ==
{| *lockAcquired*, *startSyncMeth*, *endSyncMeth*, *waitCall*, *waitRet*, *notify*, *isInterruptedCall*, *isInterruptedRet*, *interruptedCall*, *interruptedRet*, *done_toplevel_sequencer*, *get_priorityLevel* |}

2.2 MethodCallBinder

section *MethodCallBindingChannels* **parents** *scj_prelude*, *GlobalTypes*, *FrameworkChan*, *MissionId*, *MissionIds*, *SchedulableId*, *SchedulableIds*, *ThreadIds*

channel $binder\_readCall : blankID \times SchedulableID \times ThreadID$
channel $binder\_readRet : blankID \times SchedulableID \times ThreadID \times \mathbb{Z}$

$readLocs == \{BufferID\}$
$readCallers == \{ConsumerSID\}$

channel $binder\_terminationPendingCall : \times SchedulableID$
channel $binder\_terminationPendingRet : \times SchedulableID \times boolean$

$terminationPendingLocs == \{PCMissionMID\}$
$terminationPendingCallers == \{ProducerSID, ConsumerSID\}$

channel $binder\_writeCall : blankID \times SchedulableID \times ThreadID \times \mathbb{Z}$
channel $binder\_writeRet : blankID \times SchedulableID \times ThreadID$

$writeLocs == \{BufferID\}$
$writeCallers == \{ProducerSID\}$

channelset $MethodCallBinderSync == \{\!|\ done\_toplevel\_sequencer,$
$binder\_readCall, binder\_readRet,$
$binder\_terminationPendingCall, binder\_terminationPendingRet,$
$binder\_writeCall, binder\_writeRet\ |\!\}$

section *MethodCallBinder* **parents** *scj_prelude*, *MissionId*, *MissionIds*, *SchedulableId*, *SchedulableIds*, *MethodCallBindingChannels*
, *BuffeMethChan*, *PCMissionMethChan*

process $MethodCallBinder \mathrel{\widehat=}$ **begin**

$read\_MethodBinder \mathrel{\widehat=}$
$$\left(\begin{array}{l} binder\_readCall\,?\,loc : (loc \in readLocs)\,?\,caller : (caller \in readCallers)\,?\,callingThread \longrightarrow \\ readCall\,.\,loc\,.\,caller\,.\,callingThread \longrightarrow \\ readRet\,.\,loc\,.\,caller\,.\,callingThread\,?\,ret \longrightarrow \\ binder\_readRet\,.\,loc\,.\,caller\,.\,callingThread\,!\,ret \longrightarrow \\ read\_MethodBinder \end{array}\right)$$

$terminationPending\_MethodBinder \mathrel{\widehat=}$
$$\left(\begin{array}{l} binder\_terminationPendingCall\,?\,loc : (loc \in terminationPendingLocs)\,?\,caller : (caller \in terminationPendingCalle \\ terminationPendingCall\,.\,loc\,.\,caller \longrightarrow \\ terminationPendingRet\,.\,loc\,.\,caller\,?\,ret \longrightarrow \\ binder\_terminationPendingRet\,.\,loc\,.\,caller\,!\,ret \longrightarrow \\ terminationPending\_MethodBinder \end{array}\right)$$

$write\_MethodBinder \mathrel{\widehat{=}}$

$$\left(\begin{array}{l} binder\_writeCall\,?\,loc : (loc \in writeLocs)\,?\,caller : (caller \in writeCallers)\,?\,callingThread\,?\,p1 \longrightarrow \\ writeCall\,.\,loc\,.\,caller\,.\,callingThread\,!\,p1 \longrightarrow \\ writeRet\,.\,loc\,.\,caller\,.\,callingThread \longrightarrow \\ binder\_writeRet\,.\,loc\,.\,caller\,.\,callingThread \longrightarrow \\ write\_MethodBinder \end{array}\right)$$

$BinderActions \mathrel{\widehat{=}}$

$$\left(\begin{array}{l} read\_MethodBinder \\ |||\\ terminationPending\_MethodBinder \\ |||\\ write\_MethodBinder \end{array}\right)$$

- $BinderActions \triangle (done\_toplevel\_sequencer \longrightarrow \mathbf{Skip})$

end

2.3 Locking

section *NetworkLocking* **parents** *scj_prelude*, *GlobalTypes*, *FrameworkChan*, *MissionId*, *MissionIds*, *ThreadIds*, *NetworkChannels*, *ObjectFW*, *ThreadFW*, *Priority*

process $Threads \mathrel{\widehat=}$
$\left(\begin{array}{l} ThreadFW(ProducerTID, 10) \\ \vert\vert\vert \\ ThreadFW(ConsumerTID, 10) \end{array}\right)$

process $Objects \mathrel{\widehat=}$
$\big(ObjectFW(BufferOID)\big)$

process $Locking \mathrel{\widehat=} Threads \llbracket ThreadSync \rrbracket Objects$

2.4 Program

section *Program* **parents** *scj_prelude*, *MissionId*, *MissionIds*,
SchedulableId, *SchedulableIds*, *MissionChan*, *SchedulableMethChan*, *MissionFW*,
SafeletFW, *TopLevelMissionSequencerFW*, *NetworkChannels*, *ManagedThreadFW*,
SchedulableMissionSequencerFW, *PeriodicEventHandlerFW*, *OneShotEventHandlerFW*,
AperiodicEventHandlerFW, *ObjectFW*, *ThreadFW*,
PCSafeletApp, *PCMissionSequencerApp*, *PCMissionApp*, *ProducerApp*, *ConsumerApp*

process $ControlTier \mathrel{\widehat=}$
$$\left(\begin{array}{l} SafeletFW \\ \quad [\![ControlTierSync]\!] \\ TopLevelMissionSequencerFW(PCMissionSequencer) \end{array}\right)$$

process $Tier0 \mathrel{\widehat=}$
$$\left(\begin{array}{l} MissionFW(PCMissionID) \\ \quad [\![MissionSync]\!] \\ \left(\begin{array}{l} ManagedThreadFW(ProducerID) \\ \quad [\![SchedulablesSync]\!] \\ ManagedThreadFW(ConsumerID) \end{array}\right) \end{array}\right)$$

process $Framework \mathrel{\widehat=}$
$$\left(\begin{array}{l} ControlTier \\ \quad [\![TierSync]\!] \\ (\ Tier0) \end{array}\right)$$

process $Application \mathrel{\widehat=}$
$$\left(\begin{array}{l} PCSafeletApp \\ ||| \\ PCMissionSequencerApp \\ ||| \\ PCMissionApp \\ ||| \\ ProducerApp(PCMissionID) \\ ||| \\ ConsumerApp(PCMissionID) \\ ||| \\ BufferApp \end{array}\right)$$

process $Bound\_Application \mathrel{\widehat=} Application \ [\![\ MethodCallBinderSync \ ]\!] \ MethodCallBinder$
process $Program \mathrel{\widehat=} (\ Framework \ [\![\ AppSync \ ]\!] \ Bound\_Application\ ) \ [\![\ LockingSync \ ]\!] \ Locking$

3 Safelet

section $PCSafeletApp$ **parents** $scj\_prelude, SchedulableId, SchedulableIds, SafeletChan, MethodCallBindingChannels$

process $PCSafeletApp \mathrel{\widehat=}$ **begin**

$$InitializeApplication \mathrel{\widehat=} \begin{pmatrix} initializeApplicationCall \longrightarrow \\ initializeApplicationRet \longrightarrow \\ \mathbf{Skip} \end{pmatrix}$$

$$GetSequencer \mathrel{\widehat=} \begin{pmatrix} getSequencerCall \longrightarrow \\ getSequencerRet\,!\,PCMissionSequencerSID \longrightarrow \\ \mathbf{Skip} \end{pmatrix}$$

$$Methods \mathrel{\widehat=} \begin{pmatrix} GetSequencer \\ \Box \\ InitializeApplication \end{pmatrix} ;\ Methods$$

$\bullet\ (Methods) \triangle (end\_safelet\_app \longrightarrow \mathbf{Skip})$

end

4 Top Level Mission Sequencer

section *PCMissionSequencerApp* **parents** *TopLevelMissionSequencerChan*, *MissionId*, *MissionIds*, *SchedulableId*, *SchedulableIds*, *PCMissionSequencerClass*, *MethodCallBindingChannels*

process $PCMissionSequencerApp \mathrel{\widehat=} \textbf{begin}$

State

$this : \textbf{ref}\ PCMissionSequencerClass$

state *State*

Init

$State'$

$this' = \textbf{new}\ PCMissionSequencerClass()$

$GetNextMission \mathrel{\widehat=} \textbf{var}\ ret : MissionID \bullet$
$\left(\begin{array}{l} getNextMissionCall\,.\,PCMissionSequencerSID \longrightarrow \\ ret := this\,.\,getNextMission(); \\ getNextMissionRet\,.\,PCMissionSequencerSID\,!\,ret \longrightarrow \\ \textbf{Skip} \end{array}\right)$

$Methods \mathrel{\widehat=}$
$\big(GetNextMission\big)\ ;\ Methods$

$\bullet\ (Init\ ;\ Methods) \triangle (end\_sequencer\_app\,.\,PCMissionSequencerSID \longrightarrow \textbf{Skip})$

end

section *PCMissionSequencerClass* **parents** *scj_prelude*, *SchedulableId*, *SchedulableIds*, *SafeletChan*, *MethodCallBindingChannels*, *MissionId*, *MissionIds*

class $PCMissionSequencerClass \mathrel{\widehat=}$ **begin**

state *State*

$$returnedMission : \mathbb{B}$$

state *State*

initial *Init*

$$State'$$

$$returnedMission' = \textbf{False}$$

protected $getNextMission \mathrel{\widehat=}$

$$\left(\begin{array}{l}
\textbf{Skip}; \\
\textbf{if } (\neg\ returnedMission) \longrightarrow \\
\qquad \left(\begin{array}{l} \textbf{Skip}; \\ returnedMission := \textbf{True}; \\ ret := PCMissionMID \end{array}\right) \\
[]\ \neg\ (\neg\ returnedMission) \longrightarrow \\
\qquad \big(ret := nullMissionId\big) \\
\textbf{fi}
\end{array}\right)$$

- **Skip**

end

5 Missions

5.1 PCMission

section *PCMissionApp* **parents** *scj_prelude*, *MissionId*, *MissionIds*, *SchedulableId*, *SchedulableIds*, *MissionChan*, *SchedulableMethChan*, *PCMissionMethChan*, *PCMissionClass*, *MethodCallBindingChannels*

process $PCMissionApp \mathrel{\widehat=} \textbf{begin}$

State

$this : \textbf{ref}\ PCMissionClass$
$buffer : Buffer$

state *State*

Init

$State'$

$this' = \textbf{new}\ PCMissionClass()$
$buffer' =$

$$InitializePhase \mathrel{\widehat=} \begin{pmatrix} initializeCall\,.\,PCMissionMID \longrightarrow \\ register\,!\,ProducerSID\,!\,PCMissionMID \longrightarrow \\ register\,!\,ConsumerSID\,!\,PCMissionMID \longrightarrow \\ initializeRet\,.\,PCMissionMID \longrightarrow \\ \textbf{Skip} \end{pmatrix}$$

$$CleanupPhase \mathrel{\widehat=} \textbf{var}\ \mathbb{B} : ret \bullet \begin{pmatrix} cleanupMissionCall\,.\,PCMissionMID \longrightarrow \\ \begin{pmatrix} \textbf{Skip}; \\ ret := \textbf{False} \end{pmatrix} \\ cleanupMissionRet\,.\,PCMissionMID\,!\,ret \longrightarrow \\ \textbf{Skip} \end{pmatrix}$$

$$getBufferMeth \mathrel{\widehat=} \textbf{var}\ ret : Buffer \bullet \begin{pmatrix} getBufferCall\,.\,PCMissionMID \longrightarrow \\ ret := this\,.\,getBuffer(); \\ getBufferRet\,.\,PCMissionMID\,!\,ret \longrightarrow \\ \textbf{Skip} \end{pmatrix}$$

$$Methods \mathrel{\widehat=} \begin{pmatrix} InitializePhase \\ \Box \\ CleanupPhase \\ \Box \\ getBufferMeth \end{pmatrix} ;\ Methods$$

$$\bullet\ (Init\ ;\ Methods) \triangle (end\_mission\_app\,.\,PCMissionMID \longrightarrow \textbf{Skip})$$

end

section *PCMissionClass* **parents** *scj_prelude*, *SchedulableId*, *SchedulableIds*, *SafeletChan*
, *MethodCallBindingChannels*

class *PCMissionClass* $\widehat{=}$ **begin**

public *getBuffer* $\widehat{=}$
$\big($*ret* := *buffer*$\big)$

• **Skip**

end

5.2 Schedulables of PCMission

section *ProducerApp* **parents** *ManagedThreadChan*, *SchedulableId*, *SchedulableIds*, *MethodCallBindingChannels*, *MissionMethChan*, *BufferMethChan*, *ObjectIds*, *ThreadIds*

process $ProducerApp \mathrel{\widehat=}$
$pcMission : MissionID \bullet$ **begin**

State

$buffer : Buffer$

state *State*

Init

$State'$

$buffer' =$

$$
\begin{array}{l}
Run \mathrel{\widehat=} \\
\Big(runCall\,.\,ProducerSID \longrightarrow \\
\quad \Big(\mathbf{Skip}; \\
\quad \mu X \bullet \\
\quad\quad \Big(binder\_terminationPendingCall\,.\,pcMission \longrightarrow \\
\quad\quad binder\_terminationPendingRet\,.\,pcMission\,?\,terminationPending \longrightarrow \\
\quad\quad \mathbf{var}\ loopVar : \mathbb{B} \bullet loopVar := (\neg\ terminationPending); \\
\quad\quad \mathbf{if}\ (loopVar = \mathbf{True}) \longrightarrow \\
\quad\quad\quad \Big(binder\_writeCall\,.\,bufferID\,.\,ProducerSID\,.\,ProducerTID\,!\,i \longrightarrow \\
\quad\quad\quad binder\_writeRet\,.\,bufferID\,.\,ProducerSID\,.\,ProducerTID \longrightarrow \\
\quad\quad\quad \mathbf{Skip}; \\
\quad\quad\quad i := i + 1; \\
\quad\quad\quad \mathbf{var}\ keepWriting : \mathbb{B} \bullet keepWriting := this\,.\,i >= 5; \\
\quad\quad\quad \mathbf{if}\ (\neg\ keepWriting) \longrightarrow \\
\quad\quad\quad\quad \Big(requestTerminationCall\,.\,pcMission\,.\,ProducerSID \longrightarrow \\
\quad\quad\quad\quad requestTerminationRet\,.\,pcMission\,.\,ProducerSID\,?\,requestTermination \longrightarrow \\
\quad\quad\quad\quad \mathbf{Skip} \Big) \\
\quad\quad\quad [\!]\ \neg\ (\neg\ keepWriting) \longrightarrow \mathbf{Skip} \\
\quad\quad\quad \mathbf{fi} \Big) \\
\quad\quad [\!]\ (loopVar = \mathbf{False}) \longrightarrow \mathbf{Skip} \\
\quad\quad \mathbf{fi} \Big) \\
\quad ;\ X \Big) \Big) \\
\quad ; \\
runRet\,.\,ProducerSID \longrightarrow \\
\mathbf{Skip} \Big)
\end{array}
$$

$Methods \mathrel{\widehat=}$
$\big(Run\big)\ ;\ Methods$

$\bullet\ (Init\ ;\ Methods) \triangle (end\_managedThread\_app\,.\,ProducerSID \longrightarrow \mathbf{Skip})$

end

section *ProducerClass* **parents** *scj_prelude*, *SchedulableId*, *SchedulableIds*, *SafeletChan*
, *MethodCallBindingChannels*

class *ProducerClass* $\widehat{=}$ **begin**

state *State*

$i : \mathbb{Z}$

state *State*

initial *Init*

$State'$

$i' = 1$

- **Skip**

end

section *ConsumerApp* **parents** *ManagedThreadChan*, *SchedulableId*, *SchedulableIds*, *MethodCallBindingChannels*, *MissionMethChan*, *BufferMethChan*, *ObjectIds*, *ThreadIds*

process $ConsumerApp \mathrel{\widehat=}$
$pcMission : MissionID \bullet$ **begin**

State

$buffer : Buffer$

state *State*

Init

$State'$

$buffer' =$

$$
\begin{array}{l}
Run \mathrel{\widehat=} \\
\left(\begin{array}{l}
runCall\,.\,ConsumerSID \longrightarrow \\
\left(\begin{array}{l}
\textbf{Skip}; \\
\mu X \bullet \\
\left(\begin{array}{l}
binder\_terminationPendingCall\,.\,pcMission \longrightarrow \\
binder\_terminationPendingRet\,.\,pcMission\,?\,terminationPending \longrightarrow \\
\textbf{var}\ loopVar : \mathbb{B} \bullet loopVar := (\neg\ terminationPending); \\
\textbf{if}\ (loopVar = \textbf{True}) \longrightarrow \\
\quad \left(\begin{array}{l}
\textbf{var}\ result : \mathbb{Z} \bullet result := 999; \\
binder\_readCall\,.\,bufferID\,.\,ConsumerSID\,.\,ConsumerTID \longrightarrow \\
binder\_readRet\,.\,bufferID\,.\,ConsumerSID\,.\,ConsumerTID\,?\,read \longrightarrow \\
\textbf{Skip}; \\
\textbf{Skip}
\end{array}\right) ;\ X \\
[\!]\ (loopVar = \textbf{False}) \longrightarrow \textbf{Skip} \\
\textbf{fi}
\end{array}\right)
\end{array}\right) ; \\
runRet\,.\,ConsumerSID \longrightarrow \\
\textbf{Skip}
\end{array}\right)
\end{array}
$$

$Methods \mathrel{\widehat=}$
$\left(Run\right) ;\ Methods$

$\bullet\ (Init ;\ Methods) \triangle (end\_managedThread\_app\,.\,ConsumerSID \longrightarrow \textbf{Skip})$

end

section *ConsumerClass* **parents** *scj_prelude*, *SchedulableId*, *SchedulableIds*, *SafeletChan*
, *MethodCallBindingChannels*

class *ConsumerClass* $\widehat{=}$ **begin**

state *State*

$$buffer : Buffer$$

state *State*

initial *Init*

$$State'$$

- **Skip**

end