Translation Rules

BNF Encodings

section *CircusBNFEncoding* **parents** *standard_toolkit*

$[Predicate, N, Expression, Paragraph, SchemaExp, Declaration]$

$Command ::= spec \langle\!\langle \text{seq } N \times Predicate \times Predicate \rangle\!\rangle \mid equals \langle\!\langle N \times \text{seq } Expression \rangle\!\rangle$

$CParameter ::= shriek \langle\!\langle N \rangle\!\rangle \mid shriekRestrict \langle\!\langle N \times Predicate \rangle\!\rangle \mid bang \langle\!\langle Expression \rangle\!\rangle \mid$
$\quad dotParam \langle\!\langle Expression \rangle\!\rangle$

$Communication == N \times \text{seq } CParameter$

$CSExpression ::= cs \langle\!\langle \text{seq } N \rangle\!\rangle \mid csName \langle\!\langle N \rangle\!\rangle \mid$
$\quad union \langle\!\langle CSExpression \times CSExpression \rangle\!\rangle \mid$
$\quad intersect \langle\!\langle CSExpression \times CSExpression \rangle\!\rangle \mid$
$\quad subtract \langle\!\langle CSExpression \times CSExpression \rangle\!\rangle$

$Action ::= actSe \langle\!\langle SchemaExp \rangle\!\rangle \mid com \langle\!\langle Command \rangle\!\rangle \mid skip \mid stop \mid chaos \mid$
$\quad prefixExp \langle\!\langle Communication \times Action \rangle\!\rangle \mid$
$\quad guard \langle\!\langle Predicate \times Action \rangle\!\rangle \mid seqExp \langle\!\langle Action \times Action \rangle\!\rangle \mid$
$\quad extChoice \langle\!\langle Action \times Action \rangle\!\rangle \mid intChoice \langle\!\langle Action \times Action \rangle\!\rangle \mid$
$\quad actPar \langle\!\langle Action \times CSExpression \times Action \rangle\!\rangle \mid actInter \langle\!\langle Action \times Action \rangle\!\rangle \mid$
$\quad actHide \langle\!\langle Action \times CSExpression \rangle\!\rangle \mid mu \langle\!\langle N \times Action \rangle\!\rangle \mid actParam \langle\!\langle Declaration \times Action \rangle\!\rangle \mid$
$\quad actInst \langle\!\langle Action \times \text{seq } Expression \rangle\!\rangle \mid actName \langle\!\langle N \rangle\!\rangle \mid actInterupt \langle\!\langle Action \times Action \rangle\!\rangle$

$GuardedAction ::= thenAct \langle\!\langle Predicate \times Action \rangle\!\rangle \mid$
$\quad thenActComp \langle\!\langle Predicate \times Action \times GuardedAction \rangle\!\rangle$

$PParagraph ::= pPar \langle\!\langle Paragraph \rangle\!\rangle \mid actDef \langle\!\langle N \times Action \rangle\!\rangle$

$Process ::= proc \langle\!\langle \text{seq } PParagraph \times SchemaExp \times \text{seq } PParagraph \times Action \rangle\!\rangle \mid procName \langle\!\langle N \rangle\!\rangle \mid$
$\quad procSeq \langle\!\langle Process \times Process \rangle\!\rangle \mid procExtChoice \langle\!\langle Process \times Process \rangle\!\rangle \mid$
$\quad procIntChoice \langle\!\langle Process \times Process \rangle\!\rangle \mid procPar \langle\!\langle Process \times CSExpression \times Process \rangle\!\rangle \mid$
$\quad procInter \langle\!\langle Process \times Process \rangle\!\rangle \mid procHide \langle\!\langle Process \times CSExpression \rangle\!\rangle \mid$
$\quad procRename \langle\!\langle Process \times \text{seq } N \times \text{seq } N \rangle\!\rangle \mid procParam \langle\!\langle Declaration \times Process \rangle\!\rangle \mid$
$\quad procInstP \langle\!\langle Process \times \text{seq } Expression \rangle\!\rangle \mid procGeneric \langle\!\langle \text{seq } N \times Process \rangle\!\rangle \mid$
$\quad procInstG \langle\!\langle Process \times \text{seq } Expression \rangle\!\rangle \mid$
$\quad procItrInter \langle\!\langle Declaration \times Process \rangle\!\rangle$

$ProcDefinition ::= pd \langle\!\langle N \times Process \rangle\!\rangle$

$ChanSetDefinition ::= csdName \langle\!\langle N \times CSExpression \rangle\!\rangle$

$SCDeclaration ::= chanName \langle\!\langle \text{seq } N \rangle\!\rangle \mid chanNameWithType \langle\!\langle \text{seq } N \times Expression \rangle\!\rangle \mid$
$\quad scSe \langle\!\langle SchemaExp \rangle\!\rangle$

$CDeclaration ::= scDecl \langle\!\langle SCDeclaration \rangle\!\rangle \mid multiDecl \langle\!\langle SCDeclaration \times CDeclaration \rangle\!\rangle$

$ChannelDefinition == CDeclaration$

$CircusParagraph ::= para \langle\!\langle Paragraph \rangle\!\rangle \mid chanDef \langle\!\langle ChannelDefinition \rangle\!\rangle \mid$
$\quad chanSetDef \langle\!\langle ChanSetDefinition \rangle\!\rangle \mid procDef \langle\!\langle ProcDefinition \rangle\!\rangle$

$CircusProgram == \text{seq } CircusParagraph$

section *SCJBNFEncoding* **parents** *standard_toolkit*

[*MethodBody*, *ClassBodyDeclaration*, *Identifier*, *MethodDeclaration*, *Long*]

Run == *MethodBody*
ManagedThreadClassBody == *Run* × seq *ClassBodyDeclaration*
ManagedThread == *Identifier* × *ManagedThreadClassBody*

HandleAsyncEvent == *MethodBody*
HandleAsyncLongEvent == *Long* × *MethodBody*
EventHandlerClassBody == *HandleAsyncEvent* × seq *ClassBodyDeclaration*
OneShotEventHandler == *Identifier* × *EventHandlerClassBody*
LongEventHandlerClassBody == *HandleAsyncLongEvent* × seq *ClassBodyDeclaration*
AperiodicEventHandler ::= *apehType*⟪*Identifier* × *EventHandlerClassBody*⟫ |
    aplehType⟪*Identifier* × *LongEventHandlerClassBody*⟫
PeriodicEventHandler == *Identifier* × *EventHandlerClassBody*

EventHandler ::= *pehDecl*⟪*PeriodicEventHandler*⟫
    | *apehDecl*⟪*AperiodicEventHandler*⟫
    | *osehDecl*⟪*OneShotEventHandler*⟫

GetNextMission == *MethodBody*
MissionSequencerClassBody == *GetNextMission* × seq *ClassBodyDeclaration*
MissionSequencer == *Identifier* × *MissionSequencerClassBody*

NestedMissionSequencer == *MissionSequencer*

SchedulableObject ::= *handler*⟪*EventHandler*⟫
    | *mt*⟪*ManagedThread*⟫
    | *nms*⟪*NestedMissionSequencer*⟫

Cleanup == *MethodBody*
Initialize == *MethodBody*
MissionClassBody == *Initialize* × *Cleanup* × seq *ClassBodyDeclaration*
Mission == *Identifier* × *MissionClassBody*

$Cluster == Mission \times \mathbb{F} \, SchedulableObject$
$Tier == \text{seq} \, Cluster$

$TopLevelMissionSequencer ::= NoSequencer \mid tlms \langle\!\langle MissionSequencer \rangle\!\rangle$

$ImmortalMemorySize == MethodDeclaration$
$InitializeApplication == MethodBody$
$GetSequencer == MethodBody$
$SafeletClassBody ==$
$\quad InitializeApplication \times GetSequencer \times ImmortalMemorySize \times \text{seq} \, ClassBodyDeclaration$
$Safelet == Identifier \times SafeletClassBody$

$SCJProgram == Safelet \times TopLevelMissionSequencer \times \text{seq} \, Tier$

$$
\begin{array}{|l}
ProgSafelet : SCJProgram \rightarrow Safelet \\
\hline
\forall \, s : SCJProgram \\
\quad \bullet \, ProgSafelet(s) = s.1
\end{array}
$$

$$
\begin{array}{|l}
ProgTLMS : SCJProgram \rightarrow TopLevelMissionSequencer \\
\hline
\forall \, s : SCJProgram \\
\quad \bullet \, ProgTLMS(s) = s.2
\end{array}
$$

$$
\begin{array}{|l}
ProgTiers : SCJProgram \rightarrow \text{seq} \, Tier \\
\hline
\forall \, s : SCJProgram \\
\quad \bullet \, ProgTiers(s) = s.3
\end{array}
$$

Framework

section *Framework* **parents** *scj_prelude*, *SCJBNFEncoding*, *CircusBNFEncoding*

[*ID*]

[*Type*]

$$
\begin{array}{|l}
SafeletFWName : \mathbb{N} \\
TopLevelMissionSequencerFWNMame : \mathbb{N}
\end{array}
$$

$$
\begin{array}{|l}
controlTierSync : CSExpression \\
Tier0 : \mathbb{N} \\
MissionIds : \text{seq } CircusParagraph \\
SchedulableIds : \text{seq } CircusParagraph \\
ThreadIds : \text{seq } CircusParagraph \\
ObjectIds : \text{seq } CircusParagraph
\end{array}
$$

$$
\begin{array}{|l}
ServicesChan : \text{seq } CircusParagraph \\
GlobalTypes : \text{seq } CircusParagraph \\
JTime : \text{seq } CircusParagraph \\
PrimitiveTypes : \text{seq } CircusParagraph \\
Priority : \text{seq } CircusParagraph \\
PriorityQueue : \text{seq } CircusParagraph \\
FrameworkChan : \text{seq } CircusParagraph \\
MissionId : \text{seq } CircusParagraph \\
SchedulableId : \text{seq } CircusParagraph
\end{array}
$$

$$
\begin{array}{|l}
ObjectFW : CircusParagraph \\
ObjectChan : \text{seq } CircusParagraph \\
ObjectFWChan : \text{seq } CircusParagraph \\
ObjectMethChan : \text{seq } CircusParagraph \\
ThreadFW : CircusParagraph \\
ThreadChan : \text{seq } CircusParagraph \\
ThreadFWChan : \text{seq } CircusParagraph \\
ThreadMethChan : \text{seq } CircusParagraph
\end{array}
$$

$$
\begin{array}{|l}
SafeletFW : CircusParagraph \\
SafeletFWChan : \text{seq } CircusParagraph \\
SafeletChan : \text{seq } CircusParagraph \\
SafeletMethChan : \text{seq } CircusParagraph
\end{array}
$$

$TopLevelMissionSequencerFW$: $CircusParagraph$
$TopLevelMissionSequencerChan$: seq $CircusParagraph$
$TopLevelMissionSequencerFWChan$: seq $CircusParagraph$

$MissionSequencerChan$: seq $CircusParagraph$
$MissionSequencerFWChan$: seq $CircusParagraph$
$MissionSequencerMethChan$: seq $CircusParagraph$

$MissionFW$: $CircusParagraph$
$MissionChan$: seq $CircusParagraph$
$MissionFWChan$: seq $CircusParagraph$
$MissionMethChan$: seq $CircusParagraph$

$SchedulableChan$: seq $CircusParagraph$
$SchedulableMethChan$: seq $CircusParagraph$
$SchedulableFWChan$: seq $CircusParagraph$
$HandlerChan$: seq $CircusParagraph$
$HandlerFWChan$: seq $CircusParagraph$
$HandlerMethChan$: seq $CircusParagraph$

$PeriodicEventHandlerChan$: seq $CircusParagraph$
$PeriodicEventHandlerFW$: $CircusParagraph$
$PeriodicEventHandlerFWChan$: seq $CircusParagraph$
$PeriodicParameters$: seq $CircusParagraph$

$AperiodicEventHandlerChan$: seq $CircusParagraph$
$AperiodicEventHandlerFW$: $CircusParagraph$
$AperiodicLongEventHandlerMethChan$: seq $CircusParagraph$
$AperiodicParameters$: seq $CircusParagraph$

$OneShotEventHandlerChan$: seq $CircusParagraph$
$OneShotEventHandlerFW$: $CircusParagraph$
$OneShotEventHandlerFWChan$: seq $CircusParagraph$
$OneShotEventHandlerMethChan$: seq $CircusParagraph$

$SchedulableMissionSequencerFW$: $CircusParagraph$
$SchedulableMissionSequencerChan$: seq $CircusParagraph$
$SchedulableMissionSequencerFWChan$: seq $CircusParagraph$

$ManagedThreadFW$: $CircusParagraph$
$ManagedThreadChan$: seq $CircusParagraph$
$ManagedThreadFWChan$: seq $CircusParagraph$
$ManagedThreadMethChan$: seq $CircusParagraph$

$framework : CircusProgram$

$framework = ServicesChan \frown GlobalTypes \frown JTime \frown PrimitiveTypes \frown Priority \frown$
$\quad PriorityQueue \frown FrameworkChan \frown MissionId \frown SchedulableId \frown \langle ObjectFW \rangle \frown$
$\quad ObjectChan \frown ObjectFWChan \frown ObjectMethChan \frown \langle ThreadFW \rangle \frown ThreadChan \frown$
$\quad ThreadFWChan \frown ThreadMethChan \frown \langle SafeletFW \rangle \frown SafeletFWChan \frown$
$\quad SafeletChan \frown SafeletMethChan \frown \langle TopLevelMissionSequencerFW \rangle \frown$
$\quad TopLevelMissionSequencerChan \frown TopLevelMissionSequencerFWChan \frown$
$\quad MissionSequencerChan \frown MissionSequencerFWChan \frown MissionSequencerMethChan \frown$
$\quad \langle MissionFW \rangle \frown MissionChan \frown MissionFWChan \frown MissionMethChan \frown$
$\quad SchedulableChan \frown SchedulableMethChan \frown SchedulableFWChan \frown$
$\quad HandlerChan \frown HandlerFWChan \frown HandlerMethChan \frown PeriodicEventHandlerChan \frown$
$\quad \langle PeriodicEventHandlerFW \rangle \frown PeriodicEventHandlerFWChan \frown PeriodicParameters \frown$
$\quad AperiodicEventHandlerChan \frown \langle AperiodicEventHandlerFW \rangle \frown$
$\quad AperiodicLongEventHandlerMethChan \frown AperiodicParameters \frown$
$\quad OneShotEventHandlerChan \frown \langle OneShotEventHandlerFW \rangle \frown$
$\quad OneShotEventHandlerFWChan \frown OneShotEventHandlerMethChan \frown$
$\quad \langle SchedulableMissionSequencerFW \rangle \frown SchedulableMissionSequencerChan \frown$
$\quad SchedulableMissionSequencerFWChan \frown \langle ManagedThreadFW \rangle \frown ManagedThreadChan \frown$
$\quad ManagedThreadFWChan \frown ManagedThreadMethChan$

Build Phase

section *BuildPhase* **parents** *scj_prelude*, *SCJBNFEncoding*, *CircusBNFEncoding*, *Framework*

$TranslatablePrograms : \mathbb{P}\ SCJProgram$

$TranslatablePrograms \subset SCJProgram$
$\forall\, s : SCJProgram$
      • $s \in TranslatablePrograms$
          $\Leftrightarrow ProgTLMS(s) \neq NoSequencer$
          $\wedge\ ProgTiers(s) \neq \langle\rangle$
          $\wedge\ \forall\, c : Cluster$
              $\mid \exists\, t : Tier;\ tiers : \text{seq } Tier$
                  • $tiers = ProgTiers(s)$
                  $\wedge\ t \in ran\ tiers$
                  $\wedge\ c \in ran\ t$
      • $c.2 \neq \varnothing$

AppEnv
$Name : N$
$Parameters : \text{seq } Expression$

ClusterAppEnv
$Mission : AppEnv$
$Schedulables : \mathbb{F}\ AppEnv$

$Schedulables \neq \varnothing$

TierAppEnv
$Clusters : \text{seq } ClusterAppEnv$

$Clusters \neq \langle\rangle$

AppProcEnv
$Safelet : AppEnv$
$TopLevelMS : AppEnv$
$Tiers : \text{seq } TierAppEnv$

$Tiers \neq \langle\rangle$

$GetSafeletAppEnv : AppProcEnv \rightarrow AppEnv$

$\forall\, a : AppProcEnv\ \bullet$
    $GetSafeletAppEnv(a) = a.Safelet$

$GetTLMSAppEnv : AppProcEnv \rightarrow AppEnv$

$\forall\, a : AppProcEnv \bullet$
$\quad GetTLMSAppEnv(a) = a.TopLevelMS$

$GetTiersAppEnv : AppProcEnv \rightarrow \text{seq } TierAppEnv$

$\forall\, a : AppProcEnv \bullet$
$\quad GetTiersAppEnv(a) = a.Tiers$

$IDof : Identifier \nrightarrow \mathbb{N}$

$ParamsOf : \text{seq } ClassBodyDeclaration \nrightarrow \text{seq } Expression$

$BuildSOAppEnv : \mathbb{F}\, SchedulableObject \nrightarrow \mathbb{F}\, AppEnv$

$\forall\, scheds : \text{dom } BuildSOAppEnv$
$\quad |\; scheds \neq \varnothing$
$\quad \bullet\; \exists\, manT : ManagedThread;\; nestMS : NestedMissionSequencer;\; eh : EventHandler$
$\qquad perEH : PeriodicEventHandler;\; oneEH : OneShotEventHandler;$
$\qquad apehShort : Identifier \times EventHandlerClassBody;$
$\qquad apehLong : Identifier \times LongEventHandlerClassBody$
$\qquad \bullet\; BuildSOAppEnv(scheds) = \{\, a : AppEnv$
$\qquad\quad |\; \forall\, so : scheds \bullet \exists\, name : \mathbb{N};\; params : \text{seq } Expression$
$\qquad\qquad |\; so = mt(manT) \Rightarrow$
$\qquad\qquad\quad name = IDof(manT.1) \land params = ParamsOf(manT.2.2)$
$\qquad\qquad \land\; so = nms(nestMS) \Rightarrow$
$\qquad\qquad\quad name = IDof(nestMS.1) \land params = ParamsOf(nestMS.2.2)$
$\qquad\qquad \land\; so = handler(pehDecl(perEH)) \Rightarrow$
$\qquad\qquad\quad name = IDof(perEH.1) \land params = ParamsOf(perEH.2.2)$
$\qquad\qquad \land\; so = handler(osehDecl(oneEH)) \Rightarrow$
$\qquad\qquad\quad name = IDof(oneEH.1) \land params = ParamsOf(oneEH.2.2)$
$\qquad\qquad \land\; so = handler(apehDecl(apehType(apehShort))) \Rightarrow$
$\qquad\qquad\quad name = IDof(apehShort.1) \land params = ParamsOf(apehShort.2.2)$
$\qquad\qquad \land\; so = handler(apehDecl(aplehType(apehLong))) \Rightarrow$
$\qquad\qquad\quad name = IDof(apehLong.1) \land params = ParamsOf(apehLong.2.2)$
$\qquad\quad \bullet\; a = \langle\!| Name == name, Parameters == params |\!\rangle\}$

$BuildClusterAppEnv : Cluster \nrightarrow ClusterAppEnv$

$\forall\, c : \text{dom } BuildClusterAppEnv$
$\quad |\; c.2 \neq \varnothing$
$\quad \bullet\; \exists\, m : Mission;\; seqSO : \mathbb{F}\, SchedulableObject$
$\qquad |\; c = (m, seqSO)$
$\qquad \bullet\; BuildClusterAppEnv(c) =$
$\qquad\quad \langle\!| Mission == \langle\!| Name == IDof(m.1), Parameters == ParamsOf(m.2.3) |\!\rangle,$
$\qquad\qquad Schedulables == BuildSOAppEnv(seqSO) |\!\rangle$

$BuildClusterAppEnvs : \text{seq } Cluster \nrightarrow \text{seq } ClusterAppEnv$

$BuildTierAppEnv : Tier \nrightarrow TierAppEnv$

$\forall\, tier : dom\ BuildTierAppEnv$
$\quad |\ tier \neq \langle\rangle$
$\quad\quad \bullet\ BuildTierAppEnv(tier) = \langle\!| Clusters == BuildClusterAppEnvs(tier) |\!\rangle$

$BuildTiersAppEnv : seq\ Tier \nrightarrow seq\ TierAppEnv$

$\forall\, tiers : dom\ BuildTiersAppEnv$
$\quad |\ tiers \neq \langle\rangle$
$\quad\quad \bullet\ \#\,tiers = 1 \Rightarrow BuildTiersAppEnv(tiers) = \langle BuildTierAppEnv(head\ tiers)\rangle$
$\quad\quad \wedge\ \#\,tiers \geq 1 \Rightarrow BuildTiersAppEnv(tiers) =$
$\quad\quad\quad \langle BuildTierAppEnv(head\ tiers)\rangle \frown BuildTiersAppEnv(tail\ tiers)$

$BuildAppProcEnv : SCJProgram \nrightarrow AppProcEnv$

$\forall\, scjProg : dom\ BuildAppProcEnv$
$\quad |\ scjProg \in TranslatablePrograms$
$\quad\quad \bullet\ \exists\, safelet : Safelet;\ tiers : seq\ Tier;\ ms : MissionSequencer$
$\quad\quad\quad |\ scjProg = (safelet, tlms(ms), tiers)$
$\quad\quad\quad\quad \bullet\ \exists\, sfEnv : AppEnv;\ tlmsEnv : AppEnv;$
$\quad\quad\quad\quad\quad tiersEnv : seq\ TierAppEnv$
$\quad\quad\quad\quad\quad\quad \bullet\ sfEnv = \langle\!| Name == IDof(safelet.1),$
$\quad\quad\quad\quad\quad\quad\quad Parameters == ParamsOf(safelet.2.4) |\!\rangle$
$\quad\quad\quad\quad\quad\quad \wedge\ tlmsEnv = \langle\!| Name == IDof(ms.1),$
$\quad\quad\quad\quad\quad\quad\quad Parameters == ParamsOf(ms.2.2) |\!\rangle$
$\quad\quad\quad\quad\quad\quad \wedge\ tiersEnv = BuildTiersAppEnv(tiers)$
$\quad\quad\quad\quad\quad\quad \wedge\ BuildAppProcEnv(scjProg) = \langle\!| Safelet == sfEnv,$
$\quad\quad\quad\quad\quad\quad\quad TopLevelMS == tlmsEnv, Tiers == tiersEnv |\!\rangle$

$\underline{LockingEnv}$
$\quad Threads : seq(ThreadIds \times Priority)$
$\quad Objects : seq\ ObjectIds$
$\quad$
$\quad Threads \neq \langle\rangle$
$\quad Objects \neq \langle\rangle$

$BuildLockEnv : SCJProgram \nrightarrow LockingEnv$

$\forall\, scjProg : dom\ BuildLockEnv$
$\quad |\ scjProg \in TranslatablePrograms$
$\quad\quad \bullet\ \exists\, lockEnv : LockingEnv$
$\quad\quad\quad \bullet\ BuildLockEnv(scjProg) = lockEnv$

_ _ ClusterEnv _____
  $Mission : Identifier$
  $NestedMissionSequencers : \mathbb{F}\ Identifier$
  $ManagedThreads : \mathbb{F}\ Identifier$
  $PeriodicEventHandlers : \mathbb{F}\ Identifier$
  $AperiodicEventHandlers : \mathbb{F}\ Identifier$
  $OneShotEventHandlers : \mathbb{F}\ Identifier$
_____
  $disjoint\langle NestedMissionSequencers, ManagedThreads, PeriodicEventHandlers,$
       $AperiodicEventHandlers, OneShotEventHandlers\rangle$
       $\bigcup\{NestedMissionSequencers, ManagedThreads, PeriodicEventHandlers,$
       $AperiodicEventHandlers, OneShotEventHandlers\} \neq \varnothing$
_____

_ _ TierEnv _____
  $Clusters : \text{seq}\ ClusterEnv$
_____
  $Clusters \neq \langle\rangle$
_____

_ _ FWEnv _____
  $TopLevelMS : Identifier$
  $Tiers : \text{seq}\ TierEnv$
_____
  $Tiers \neq \langle\rangle$
_____

  $GetTierFWEnvs : FWEnv \rightarrow \text{seq}\ TierEnv$
_____
  $\forall\, env : FWEnv$
       $\bullet\ GetTierFWEnvs(env) = env.Tiers$
_____

  $GetIdentifiers : \mathbb{F}\ SchedulableObject \nrightarrow \mathbb{F}\ Identifier$
_____
  $\forall\, scheds : dom\ GetIdentifiers$
       $\mid scheds \neq \varnothing$
       $\bullet\ \exists\, manT : ManagedThread;\ nestMS : NestedMissionSequencer;$
            $perEH : PeriodicEventHandler;\ oneEH : OneShotEventHandler;$
            $eh : EventHandler;$
            $apehShort : Identifier \times EventHandlerClassBody;$
            $apehLong : Identifier \times LongEventHandlerClassBody$
                 $\bullet\ GetIdentifiers(scheds) = \{i : Identifier$
                      $\mid \forall\, s : scheds$
                      $\mid scheds \neq \varnothing$
                      $\bullet\ s = mt(manT) \Rightarrow i = manT.1$
                      $\wedge\ s = nms(nestMS) \Rightarrow i = nestMS.1$
                      $\wedge\ s = handler(pehDecl(perEH)) \Rightarrow i = perEH.1$
                      $\wedge\ s = handler(apehDecl(apehType(apehShort))) \Rightarrow i = apehShort.1$
                      $\wedge\ s = handler(apehDecl(aplehType(apehLong))) \Rightarrow i = apehLong.1$
                      $\wedge\ s = handler(osehDecl(oneEH)) \Rightarrow i = oneEH.1$
                 $\}$
_____

$BuildSOEnvs : \mathbb{F}\, SchedulableObject \nrightarrow$
$\quad \mathbb{F}\, Identifier \times \mathbb{F}\, Identifier \times \mathbb{F}\, Identifier \times$
$\quad \mathbb{F}\, Identifier \times \mathbb{F}\, Identifier$

$\forall\, s : dom\ BuildSOEnvs$
$\quad |\ s \neq \varnothing$
$\quad \bullet\ \exists\, sms : \mathbb{F}\, Identifier;\ pehs : \mathbb{F}\, Identifier;$
$\qquad\quad apehs : \mathbb{F}\, Identifier;\ osehs : \mathbb{F}\, Identifier;\ mts : \mathbb{F}\, Identifier$
$\qquad |\ mts = GetIdentifiers(\{mtSched : s$
$\qquad\quad |\ \exists\, m : ManagedThread$
$\qquad\quad \bullet\ mtSched = mt(m)\})$
$\qquad \wedge\ sms = GetIdentifiers(\{nmsSched : s$
$\qquad\quad |\ \exists\, n : NestedMissionSequencer$
$\qquad\quad \bullet\ nmsSched = nms(n)\})$
$\qquad \wedge\ pehs = GetIdentifiers(\{pehSched : s$
$\qquad\quad |\ \exists\, p : PeriodicEventHandler$
$\qquad\quad \bullet\ pehSched = handler(pehDecl(p))\})$
$\qquad \wedge\ apehs = GetIdentifiers(\{apehSched : s$
$\qquad\quad |\ \exists\, a : Identifier \times EventHandlerClassBody$
$\qquad\quad \bullet\ apehSched = handler(apehDecl(apehType(a)))\})$
$\qquad \wedge\ apehs = GetIdentifiers(\{apehLSched : s$
$\qquad\quad |\ \exists\, a : Identifier \times LongEventHandlerClassBody$
$\qquad\quad \bullet\ apehLSched = handler(apehDecl(aplehType(a)))\})$
$\qquad \wedge\ osehs = GetIdentifiers(\{osehSched : s$
$\qquad\quad |\ \exists\, o : OneShotEventHandler$
$\qquad\quad \bullet\ osehSched = handler(osehDecl(o))\})$
$\quad \bullet\ BuildSOEnvs(s) = (sms, pehs, apehs, osehs, mts)$

$BuildClusterEnv : Cluster \nrightarrow ClusterEnv$

$\forall\, c : dom\ BuildClusterEnv$
$\quad |\ c.2 \neq \varnothing$
$\quad \bullet\ \exists\, missionName : Identifier;\ sms : \mathbb{F}\, Identifier;\ pehs : \mathbb{F}\, Identifier;$
$\qquad\quad apehs : \mathbb{F}\, Identifier;\ oseh : \mathbb{F}\, Identifier;\ mts : \mathbb{F}\, Identifier;\ cluster : ClusterEnv$
$\quad |\ missionName = c.1.1$
$\quad \wedge\ (sms, pehs, apehs, oseh, mts) = BuildSOEnvs(c.2)$
$\quad \bullet\ BuildClusterEnv(c) =$
$\qquad \langle\!| Mission == missionName, NestedMissionSequencers == sms, PeriodicEventHandler$
$\qquad\ AperiodicEventHandlers == apehs, OneShotEventHandlers == oseh, ManagedThreads$

$BuildClusterEnvs : seq\ Cluster \nrightarrow seq\ ClusterEnv$

$\forall\, c : dom\ BuildClusterEnvs$
$\quad |\ c \neq \langle\rangle \wedge \forall\, s : seq\ Cluster \bullet s \neq \langle\rangle$
$\quad \bullet\ \#\, c = 1 \Rightarrow BuildClusterEnvs(c) = \langle BuildClusterEnv(head\ c)\rangle$
$\quad \wedge\ \#\, c \geq 1 \Rightarrow BuildClusterEnvs(c) = \langle BuildClusterEnv(head\ c)\rangle \frown BuildClusterEnvs(tail\ c)$

$BuildTierEnv : Tier \nrightarrow TierEnv$

$\forall\, tier : seq\ Cluster$
$\quad \bullet\ BuildTierEnv(tier) = \langle\!| Clusters == BuildClusterEnvs(tier)|\!\rangle$

$\quad$ _BuildTierEnvs_ : seq _Tier_ $\nrightarrow$ seq _TierEnv_
$\rule{8cm}{0.4pt}$
$\quad \forall\, tiers$: seq _Tier_ $\bullet$
$\qquad BuildTierEnvs(tiers) = \langle BuildTierEnv(head\ tiers)\rangle \frown BuildTierEnvs(tail\ tiers)$

$\quad$ _BuildFWEnv_ : _SCJProgram_ $\nrightarrow$ _FWEnv_
$\rule{8cm}{0.4pt}$
$\quad \forall\, scjProg$: _dom BuildFWEnv_
$\qquad |\ scjProg \in TranslatablePrograms$
$\qquad \bullet \exists\, tlms$: _MissionSequencer_; $tlmsID$: _Identifier_; $tlmsBody$: _MissionSequencerClassBody_;
$\qquad\qquad tiers$: seq _Tier_ $|$
$\qquad\qquad\quad scjProg.2 \neq NoSequencer \Rightarrow tlms = (tlmsID, tlmsBody)$
$\qquad\qquad\quad \wedge\ tiers = scjProg.3 \bullet$
$\qquad\qquad\quad BuildFWEnv(scjProg) =$
$\qquad\qquad\qquad \langle\!\vert\, TopLevelMS == tlms.1,\ Tiers == BuildTierEnvs(tiers)\,\vert\!\rangle$

$\rule{10cm}{0pt}$

$\underline{\quad BinderMethodEnv\ }\rule{8cm}{0.4pt}$
$\quad MethodName : N$
$\quad Locs : \mathbb{F}\, N$
$\quad Callers : \mathbb{F}\, N$
$\quad ReturnType : Type$
$\quad Params : $ seq $Type$
$\quad Synchrinised : \mathbb{B}$
$\quad LocParam : N$
$\quad LocType : Type$
$\quad CallerType : Type$
$\rule{10cm}{0.4pt}$

$\underline{\quad MCBEnv\ }\rule{8cm}{0.4pt}$
$\quad BinderMethods : $ seq $BinderMethodEnv$
$\rule{8cm}{0.4pt}$
$\quad BinderMethods \neq \langle\rangle$
$\rule{10cm}{0.4pt}$

$\quad$ _GetSFMethods_ : _Safelet_ $\rightarrow$ seq _ClassBodyDeclaration_
$\rule{8cm}{0.4pt}$
$\quad \forall\, sf$: _Safelet_
$\qquad \bullet GetSFMethods(sf) = sf.2.4$

$\quad$ _GetTLMSMethods_ : _MissionSequencer_ $\rightarrow$ seq _ClassBodyDeclaration_
$\rule{8cm}{0.4pt}$
$\quad \forall\, tlms$: _MissionSequencer_
$\qquad \bullet GetTLMSMethods(tlms) = tlms.2.2$

$\quad$ _BuildBME_ : $N \times N \times$ _MethodDeclaration_ $\rightarrow$ _BinderMethodEnv_

$\quad$ _BuildSFMCBEnv_ : seq _ClassBodyDeclaration_ $\rightarrow$ seq _BinderMethodEnv_
$\rule{8cm}{0.4pt}$
$\quad \forall\, body$: _dom BuildSFMCBEnv_
$\qquad |\ body \neq \langle\rangle \wedge \forall\, b$: seq _ClassBodyDeclaration_ $\bullet\ b \neq \langle\rangle$
$\qquad \bullet BuildSFMCBEnv(body) = \langle\rangle$

$BuildTLMSMCBEnv : \text{seq } ClassBodyDeclaration \rightarrow \text{seq } BinderMethodEnv$

$BuildTierMCBEnvs : \text{seq } Tier \rightarrow \text{seq } BinderMethodEnv$

$BuildMCBEnv : SCJProgram \nrightarrow MCBEnv$

$\forall\, scjProg : dom\, BuildMCBEnv$
$\quad | \; scjProg \in TranslatablePrograms$
$\quad\quad \bullet \, \exists\, sf : Safelet;\; tlms : MissionSequencer;\; tiers : \text{seq } Tier$
$\quad\quad\quad\quad | \; sf = scjProg.1$
$\quad\quad\quad\quad \wedge \; tiers = scjProg.3$
$\quad\quad\quad\quad \bullet \, BuildMCBEnv(scjProg) =$
$\quad\quad\quad\quad\quad \langle\!| BinderMethods == BuildSFMCBEnv(GetSFMethods(sf))$
$\quad\quad\quad\quad\quad ^\frown BuildTLMSMCBEnv(GetTLMSMethods(tlms))$
$\quad\quad\quad\quad\quad ^\frown BuildTierMCBEnvs(tiers) |\!\rangle$

Generate Phase

section *GeneratePhase* **parents** *scj_prelude*, *Framework*, *BuildPhase*

$procNameOf : Process \rightarrow N$

$ControlTierSync : CSExpression$
$MissionSync : CSExpression$
$SchedulablesSync : CSExpression$

$TierSync : TierEnv \rightarrow CSExpression$

$\forall t : TierEnv$
    $\bullet \exists m : \text{seq } N$
        $\bullet TierSync(t) = cs(m)$

$GetMissionID : ClusterEnv \nrightarrow N$

$GenerateTiersFWProc : ClusterEnv \nrightarrow Process$

$GenerateClusterFWProcs : \text{seq } ClusterEnv \nrightarrow Process$

$\forall clusters : \text{dom } GenerateClusterFWProcs$
    $\mid clusters \neq \langle \rangle$
    $\bullet \# clusters = 1$
        $\Rightarrow GenerateClusterFWProcs(clusters) =$
            $procPar($
                $procName(GetMissionID(head\ clusters)),$
                $MissionSync,$
                $GenerateTiersFWProc(head\ clusters)$
            $)$
    $\wedge \# clusters \geq 1$
        $\Rightarrow GenerateClusterFWProcs(clusters) =$
            $procPar($
                $procPar($
                    $procName(GetMissionID(head\ clusters)),$
                    $MissionSync,$
                    $GenerateTiersFWProc(head\ clusters)),$
                $SchedulablesSync,$
                $GenerateClusterFWProcs(tail\ clusters)$
            $)$

$GenerateTierFWProcs : \text{seq } TierEnv \nrightarrow \text{seq } Process$

$\forall\, tiers : \text{seq } TierEnv$
$\quad | \ tiers \neq \langle\rangle$
$\quad\quad \bullet\ \#\, tiers = 1 \Rightarrow$
$\quad\quad\quad GenerateTierFWProcs(tiers) = \langle GenerateClusterFWProcs((head\ tiers).Clusters)\rangle$
$\quad\quad \wedge\ \#\, tiers \geq 1 \Rightarrow$
$\quad\quad\quad GenerateTierFWProcs(tiers) =$
$\quad\quad\quad\quad \langle GenerateClusterFWProcs((head\ tiers).Clusters)\rangle$
$\quad\quad\quad\quad {}^\frown GenerateTierFWProcs(tail\ tiers)$

$GenerateTierFWProc : \text{seq } TierEnv \nrightarrow Process$

$ControlTier : N$
$TopLevelMissionSequencerFWName : N$

$GetParams : Identifier \nrightarrow \text{seq } Expression$

$GenerateFWProcs : FWEnv \nrightarrow \text{seq } Process$

$\forall\, env : FWEnv$
$\quad | \ env.Tiers \neq \langle\rangle$
$\quad\quad \bullet\ \exists\, fwProc : Process;\ controlTierProc : Process;\ tierProcs : \text{seq } Process$
$\quad\quad\quad | \ fwProc = procPar($
$\quad\quad\quad\quad procName(ControlTier),$
$\quad\quad\quad\quad TierSync(head\ env.Tiers),$
$\quad\quad\quad\quad GenerateTierFWProc(env.Tiers)$
$\quad\quad\quad)$
$\quad\quad\quad \wedge\ controlTierProc = procPar($
$\quad\quad\quad\quad procName(SafeletFWName),$
$\quad\quad\quad\quad ControlTierSync,$
$\quad\quad\quad\quad procInstP(procName(TopLevelMissionSequencerFWName), GetParams(env.TopL$
$\quad\quad\quad)$
$\quad\quad\quad \wedge\ tierProcs = GenerateTierFWProcs(env.Tiers)$
$\quad\quad\quad \bullet\ GenerateFWProcs(env) = \langle fwProc\rangle {}^\frown \langle controlTierProc\rangle {}^\frown tierProcs$

$GenerateAppTierProcs : \text{seq } TierAppEnv \nrightarrow Process$

$GenerateAppProc : AppProcEnv \nrightarrow Process$

$\forall\, appProcEnv : AppProcEnv$
$\quad\quad \bullet\ \exists\, sfAppEnv : AppEnv;\ tlmsAppEnv : AppEnv;\ tiersAppEnvs : \text{seq } TierAppEnv$
$\quad\quad\quad | \ sfAppEnv = GetSafeletAppEnv(appProcEnv)$
$\quad\quad\quad \wedge\ tlmsAppEnv = GetTLMSAppEnv(appProcEnv)$
$\quad\quad\quad \wedge\ tiersAppEnvs = GetTiersAppEnv(appProcEnv)$
$\quad\quad\quad \bullet\ GenerateAppProc(appProcEnv) =$
$\quad\quad\quad\quad procInter($
$\quad\quad\quad\quad\quad procInter($
$\quad\quad\quad\quad\quad\quad procInstP(procName(sfAppEnv.Name), sfAppEnv.Parameters),$
$\quad\quad\quad\quad\quad\quad procInstP(procName(tlmsAppEnv.Name), tlmsAppEnv.Parameters)$
$\quad\quad\quad\quad\quad),$
$\quad\quad\quad\quad GenerateAppTierProcs(tiersAppEnvs)$
$\quad\quad\quad\quad)$

$Locking : N$
$Threads : N$
$ThreadSync : CSExpression$
$Objects : N$

$BinderCallChan : N \nrightarrow \text{seq } N$

$NaturalCallChan : N \nrightarrow \text{seq } N$

$NaturalRetChan : N \nrightarrow \text{seq } N$

$BindeRetChan : N \nrightarrow \text{seq } N$

$MCBParams : \text{seq } Type \nrightarrow Expression$

$GenerateMCBChan : BinderMethodEnv \rightarrow CircusParagraph$

$\forall\, bme : BinderMethodEnv$
$\quad \bullet\; GenerateMCBChan(bme) = chanDef($
$\qquad multiDecl(chanNameWithType(NaturalCallChan(bme.MethodName),$
$\qquad\qquad MCBParams(bme.Params)),$
$\qquad\quad scDecl(chanNameWithType(NaturalRetChan(bme.MethodName),$
$\qquad\qquad MCBParams(bme.Params))))$
$\quad)$

$MethodCallBinderSync : N$
$GenerateMethodCallBinderSync : \text{seq } BinderMethodEnv \nrightarrow CircusParagraph$

$GenerateMCBChans : \text{seq } BinderMethodEnv \rightarrow \text{seq } CircusParagraph$

$\forall\, bEnvs : \text{seq } BinderMethodEnv$
$\quad |\; bEnvs \neq \langle\rangle$
$\quad \bullet\; \#\, bEnvs = 1 \Rightarrow$
$\qquad GenerateMCBChans(bEnvs) = \langle GenerateMCBChan(head\; bEnvs)\rangle$
$\quad \wedge\; \#\, bEnvs \geq 1 \Rightarrow$
$\qquad GenerateMCBChans(bEnvs) = \langle GenerateMCBChan(head\; bEnvs)\rangle$
$\qquad\quad \frown GenerateMCBChans(tail\; bEnvs)$

$BinderCallComm : N \nrightarrow N$

$NaturalCallComm : N \nrightarrow N$

$NaturalRetComm : N \nrightarrow N$

$BindeRetComm : N \nrightarrow N$

$GenerateMCBName : N \nrightarrow N$

$BinderCallParams : \text{seq } Type \rightarrow \text{seq } CParameter$

$NaturalCallParams : \text{seq } Type \rightarrow \text{seq } CParameter$

$NaturalRetParams : \text{seq } Type \rightarrow \text{seq } CParameter$

$BinderRetParams : \text{seq } Type \rightarrow \text{seq } CParameter$

$BinderActions : N$
$DoneTLS : Communication$
$NoState : SchemaExp$
$MethodCallBinder : N$

$GenerateMCBAction : BinderMethodEnv \rightarrow PParagraph$

$\forall\, bme : BinderMethodEnv$
    $\bullet\ GenerateMCBAction(bme) = actDef(GenerateMCBName(bme.MethodName),$
        $prefixExp((BinderCallComm(bme.MethodName),$
            $BinderCallParams(bme.Params)),$
          $prefixExp((NaturalCallComm(bme.MethodName),$
             $BinderCallParams(bme.Params)),$
            $prefixExp((NaturalRetComm(bme.MethodName),$
               $BinderCallParams(bme.Params)),$
              $prefixExp((BindeRetComm(bme.MethodName),$
                 $BinderCallParams(bme.Params)),$
                $actName(GenerateMCBName(bme.MethodName))$
               $)$
             $)$
           $)$
         $)$
      $)$

$GenerateMCBActions : \text{seq } BinderMethodEnv \nrightarrow \text{seq } PParagraph$

$\forall bEnvs : \text{seq } BinderMethodEnv$
$\quad | \ bEnvs \neq \langle\rangle$
$\quad\quad \bullet \ \# bEnvs = 1 \Rightarrow$
$\quad\quad\quad GenerateMCBActions(bEnvs) = \langle GenerateMCBAction(head\ bEnvs)\rangle$
$\quad\quad \wedge \ \# bEnvs \geq 1 \Rightarrow$
$\quad\quad\quad GenerateMCBActions(bEnvs) = \langle GenerateMCBAction(head\ bEnvs)\rangle$
$\quad\quad\quad\quad \frown GenerateMCBActions(tail\ bEnvs)$

$GenerateMCBProc : \text{seq } BinderMethodEnv \nrightarrow CircusParagraph$

$\forall bmes : \text{seq } BinderMethodEnv$
$\quad | \ bmes \neq \langle\rangle$
$\quad\quad \bullet \ GenerateMCBProc(bmes) =$
$\quad\quad\quad procDef(pd(MethodCallBinder,$
$\quad\quad\quad\quad proc($
$\quad\quad\quad\quad\quad \langle\rangle,$
$\quad\quad\quad\quad\quad NoState,$
$\quad\quad\quad\quad\quad GenerateMCBActions(bmes),$
$\quad\quad\quad\quad\quad actInterupt(actName(BinderActions), prefixExp(DoneTLS, skip))$
$\quad\quad\quad\quad)$
$\quad\quad\quad))$

$GenerateMCBModel : MCBEnv \nrightarrow \text{seq } CircusParagraph$

$\forall bEnv : MCBEnv$
$\quad\quad \bullet \ GenerateMCBModel(bEnv) = GenerateMCBChans(bEnv.BinderMethods)\frown$
$\quad\quad\quad \langle GenerateMethodCallBinderSync(bEnv.BinderMethods), GenerateMCBProc(bEnv.Bin$

$GenerateThreadProc : \text{seq}(ThreadIds \times Priority) \nrightarrow Process$

$GenerateObjectProc : \text{seq } ObjectIds \nrightarrow Process$

$GenerateLockModel : LockingEnv \nrightarrow \text{seq } CircusParagraph$

$\forall lEnv : dom\ GenerateLockModel$
$\quad | \ lEnv.Threads \neq \langle\rangle \wedge lEnv.Objects \neq \langle\rangle$
$\quad\quad \bullet \ GenerateLockModel(lEnv) =$
$\quad\quad\quad \langle$
$\quad\quad\quad\quad procDef(pd(Locking, procPar(procName(Threads),$
$\quad\quad\quad\quad\quad ThreadSync,$
$\quad\quad\quad\quad\quad procName(Objects)))$
$\quad\quad\quad\quad),$
$\quad\quad\quad\quad procDef(pd(Threads, GenerateThreadProc(lEnv.Threads))),$
$\quad\quad\quad\quad procDef(pd(Objects, GenerateObjectProc(lEnv.Objects)))$
$\quad\quad\quad \rangle$

Translate SCJ Program

section *TransSCJProg* **parents** *scj_prelude*, *SCJBNFEncoding*, *CircusBNFEncoding*,
  BuildPhase, *GeneratePhase*, *Framework*

$ProcessID : \mathbb{N} \nrightarrow ID$

$TransClasses : SCJProgram \nrightarrow CircusProgram$

$FWName : \mathbb{N}$
$AppName : \mathbb{N}$
$MCBName : \mathbb{N}$
$LockName : \mathbb{N}$
$ProgName : Identifier \nrightarrow \mathbb{N}$

$appComms : CSExpression$
$mcbComms : CSExpression$
$lockComms : CSExpression$

$TransSCJProg : Identifier \times SCJProgram \nrightarrow CircusProgram$

$\forall\, name : Identifier;\ scjProg : SCJProgram$
$\quad |\ (name, scjProg) \in dom\ TransSCJProg\ \bullet$
$\quad \exists\, app : CircusProgram;\ program : CircusProgram;$
$\qquad\qquad fwProcs : seq\ Process;\ appProc : Process;\ lockModel : seq\ CircusParagraph;$
$\qquad mcbModel : seq\ CircusParagraph;$
$\qquad fwEnv : FWEnv;\ appEnv : AppProcEnv;\ mcbEnv : MCBEnv;\ lockEnv : LockingEnv\ |$
$\qquad\quad fwEnv = BuildFWEnv(scjProg)$
$\qquad\quad appEnv = BuildAppProcEnv(scjProg)$
$\qquad\quad mcbEnv = BuildMCBEnv(scjProg)$
$\qquad\quad lockEnv = BuildLockEnv(scjProg)$
$\qquad\quad app = TransClasses(scjProg)\ \wedge$
$\qquad\quad fwProcs = GenerateFWProcs(fwEnv)\ \wedge$
$\qquad\quad appProc = GenerateAppProc(appEnv)\ \wedge$
$\qquad\quad mcbModel = GenerateMCBModel(mcbEnv)\ \wedge$
$\qquad\quad lockModel = GenerateLockModel(lockEnv)\ \wedge$
$\qquad\quad program = \langle procDef(pd(ProgName(name),$
$\qquad\quad procHide(procPar($
$\qquad\qquad procHide($
$\qquad\qquad\quad procPar($
$\qquad\qquad\qquad procName(FWName),$
$\qquad\qquad\qquad appComms,$
$\qquad\qquad\qquad procHide($
$\qquad\qquad\qquad\quad procPar(procName(AppName),$
$\qquad\qquad\qquad\quad mcbComms,$
$\qquad\qquad\qquad\quad procName(MCBName)),$
$\qquad\qquad\qquad mcbComms)),$
$\qquad\qquad\quad appComms),$
$\qquad\qquad lockComms,$
$\qquad\qquad procName(LockName)),$
$\qquad\quad lockComms)))\rangle\ \bullet$
$\qquad\quad TransSCJProg(name, scjProg) =$
$\qquad\qquad framework\ ^\frown \langle procDef(pd(FWName, head\ fwProcs))\rangle ^\frown$
$\qquad\qquad\quad app\ ^\frown \langle procDef(pd(AppName, appProc))\rangle ^\frown$
$\qquad\qquad\quad mcbModel ^\frown$
$\qquad\qquad\quad lockModel ^\frown$
$\qquad\qquad\quad program$

Low Level

- *Method* : *MethodDeclaration* ⇸ (*Name*, *Params*, *ReturnType*, *Body*) : translates an active application method into a **Circus** action

- *DataMethod* : *MethodDeclaration* ⇸ : translates data methods into an **Oh Circus** method

- *MethodBody* : *Block* ⇸ seq *CircExpression* : translates a Java block, for example a method body

- *Registers* : *Block* ⇸ seq *Name* : extracts the Names of the schedulables registered in a Java block

- *Returns* : *Block* ⇸ seq *Name* : extracts the Names of the variables retuned in a Java block

- *Variable* : (*Name*, *Type*, *InitExpression*) ⇸ (*CircName*, *CircType*, *CircExpression*) : translates a variable

- *Parameters* : (*Name*, *Params*, *ReturnType*, *Body*) ⇸ seq *CircParam* : translates a list of method parameters

- ⟦*Name*⟧$_{name}$: translates the *name* to a Z identifier

- ⟦*varType*⟧$_{type}$: translates types

- ⟦*expr*⟧$_{expression}$: translates expressions

Auxiliary Functions

- *IdOf*(*name*): yields the identifier of a component called *name*

- *ObjectIdOf*(*name*): yields the identifier of the *Object* process of a component called *name*

- *ThreadIdOf*(*name*): yields the identifier of the *Thread* process of a component called *name*

- *MethodName*(*method*): yields the method name of *method*

- *MethodsOf*(*name*) : yeilds a sequence of methods from the class *name*

Pattern Matching Rules

Safelet

```
 1  public class Identifier implements Safelet
 2  {
 3     FieldDeclaration_1
 4     . . .
 5     FieldDeclaration_n
 6
 7     ConstructorDeclaration
 8
 9     initializeApplication
10
11     getSequencer
12
13     AppMeth_1
14     . . .
15     AppMeth_n
16  }
```

$$\mathbf{process} \, [\![Identifier]\!]_{Name} \, App \mathrel{\widehat{=}} [\![\, [\![ConstructorDeclaration]\!]_{Method}]\!]_{Parameters} \, \mathbf{begin}$$

State
    $this : \mathrm{ref} \, [\![Identifier]\!]_{name} \, Class$
---

state *State*

Init
    $State'$
    $this := \mathbf{new} \, [\![Identifier]\!]_{name} \, Class()$

$$InitializeApplication \mathrel{\widehat{=}}$$
$$\begin{pmatrix} initializeApplicationCall \longrightarrow \\ [\![\, [\![InitializeApplication]\!]_{Method}]\!]_{MethBody} \\ initializeApplicationRet \longrightarrow \\ \mathbf{Skip} \end{pmatrix}$$

$$GetSequencer \mathrel{\widehat{=}}$$
$$\begin{pmatrix} getSequencerCall \longrightarrow \\ getSequencerRet \, ! \, [\![GetSequencer]\!]_{Returns} \longrightarrow \\ \mathbf{Skip} \end{pmatrix}$$

$$[\![AppMeth\_1]\!]_{Method}$$

$$\ldots$$

$$[\![AppMeth\_n]\!]_{Method}$$

$Methods \mathrel{\widehat{=}}$

$$
\begin{pmatrix}
GetSequencer \\
\square \\
InitializeApplication \\
\square \\
MethName(AppMeth\_1) \\
\square \\
\dots \\
MethName(AppMeth\_n) \\
\dots
\end{pmatrix}
; \quad Methods
$$

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

end

Mission Sequencer

```
1  public class Identifier extends MissionSequencer
2  {
3      FieldDeclaration_1
4      . . .
5      FieldDeclaration_n
6
7      ConstructorDeclaration
8
9      getNextMission
10
11     AppMeth_1
12     . . .
13     AppMeth_n
14 }
```

$\mathbf{process}\, [\![\, Identifier\,]\!]_{Name}\; App \mathrel{\widehat{=}} [\![\, [\![\, ConstructorDeclaration\,]\!]_{Method}\,]\!]_{Parameters}\; \mathbf{begin}$

$State$
$\quad this : \mathrm{ref}\, [\![\, Identifier\,]\!]_{name}\; Class$

$\mathbf{state}\, State$

$Init$
$\quad State'$
$\quad this := \mathbf{new}\, [\![\, Identifier\,]\!]_{name}\; Class()$

$GetNextMission \mathrel{\widehat{=}} \mathbf{var}\, ret : MissionID \bullet$
$\begin{pmatrix} getNextMissionCall\, .\, IdOf(Identifier) \longrightarrow \\ ret := this\, .\, getNextMission(); \\ getNextMissionRet\, .\, IdOf(Identifier)\, !\, ret \longrightarrow \\ \mathbf{Skip} \end{pmatrix}$

$[\![AppMeth\_1]\!]_{Method}$

$\quad . . .$

$[\![AppMeth\_n]\!]_{Method}$

$Methods \mathrel{\widehat{=}}$
$\begin{pmatrix} GetNextMission \\ \Box \\ MethName(AppMeth\_1) \\ \Box \\ MethName(AppMeth\_n) \\ . . . \end{pmatrix}\; ;\; Methods$

$\quad \bullet\, (Init\, ;\; Methods)\; \triangle (end\_sequencer\_app\, .\, IdOf(Identifier) \longrightarrow \mathbf{Skip})$

$\mathbf{end}$

Mission

```
1  public class Identifier extends Mission
2  {
3     FieldDeclaration_1
4      . . .
5     FieldDeclaration_n
6
7     ConstructorDeclaration
8
9     initialize
10
11    cleanUp
12
13    AppMeth_1
14     . . .
15    AppMeth_n
16 }
```

$$\textbf{process}\, [\![\, Indentifier\,]\!]\, App \,\widehat{=}\, [\![\, [\![\, ConstructorDeclaration\,]\!]_{Method}\,]\!]_{Parameters}\, \textbf{begin}$$

$$\begin{array}{|l}\hline State\\\hline \quad this : \mathrm{ref}\, [\![\, Identifier\,]\!]_{name}\, Class\\\hline\end{array}$$

state $State$

$$\begin{array}{|l}\hline Init\\\hline \quad State'\\ \quad this := \textbf{new}\, [\![\, Identifier\,]\!]_{name}\, Class()\\\hline\end{array}$$

$InitializePhase \,\widehat{=}$
$$\begin{pmatrix} initializeCall\,.\,IdOf(Indentifier) \longrightarrow\\ [\![\, initialize\,]\!]_{Registers}\, initializeRet\,.\,IdOf(Indentifier) \longrightarrow\\ \textbf{Skip} \end{pmatrix}$$

$CleanupPhase \,\widehat{=}$
$$\begin{pmatrix} cleanupMissionCall\,.\,IdOf(Indentifier) \longrightarrow\\ cleanupMissionRet\,.\,IdOf(Indentifier)\,!\,\textbf{True} \longrightarrow\\ \textbf{Skip} \end{pmatrix}$$

$[\![\, AppMeth\_1]\!]_{Method}$

$\quad \dots$

$[\![\, AppMeth\_n]\!]_{Method}$

$$Methods \,\widehat{=}\, \begin{pmatrix} InitializePhase\\ \square\\ CleanupPhase\\ \square\\ MethName(AppMeth\_1)\\ \square\\ MethName(AppMeth\_n)\\ \dots \end{pmatrix} ;\ Methods$$

$\bullet\ (\mathit{Init}\ ;\ \ \mathit{Methods})\quad \triangle(\mathit{end\_mission\_app}\,.\,\mathit{IdOf}\,(\mathit{Identifier}) \longrightarrow \mathbf{Skip}$

end

Handlers

```
1  class Identifier extends HandlerType
2  {
3     FieldDeclaration_1
4     ...
5     FieldDeclaration_n
6
7     ConstructorDeclaration
8
9     handleAsyncEvent
10
11    AppMeth_1
12    ...
13    AppMeth_n
14 }
```

$$\textbf{process}\,[\![\,PName\,]\!]\,App \,\widehat{=}\, [\![\,[\![\,ConstructorDeclaration\,]\!]_{Method}\,]\!]_{Parameters}\,\textbf{begin}$$

―――― *State* ――――――――――――――――――――――――――――
$$this : \text{ref}\,[\![\,Identifier\,]\!]_{name}\,Class$$

state *State*

―――― *Init* ――――――――――――――――――――――――――――
$$State'$$
$$this := \textbf{new}\,[\![\,Identifier\,]\!]_{name}\,Class()$$

$$handleAsyncEvent \,\widehat{=}\,$$
$$\begin{pmatrix} handleAsyncEventCall\,.\,IdOf(PName) \longrightarrow \\ [\![\,[\![\,HandleAsyncBody\,]\!]_{Method}\,]\!]_{MethBody}; \\ handleAsyncEventRet\,.\,IdOf(PName) \longrightarrow \\ \textbf{Skip} \end{pmatrix}$$

$$[\![\,AppMeth\_1\,]\!]_{Method}$$

$$\ldots$$

$$[\![\,AppMeth\_n\,]\!]_{Method}$$

$$Methods \,\widehat{=}\,$$
$$\begin{pmatrix} handleAsyncEvent \\ \square \\ MethName(AppMeth\_1) \\ \square \\ MethName(AppMeth\_n) \\ \ldots \end{pmatrix} ;\;\; Methods$$

$$\bullet\,(Init;\;\,Methods)\;\,\triangle(end\_[\![\,HandlerTypeIdOf(PName)\,]\!]\longrightarrow\textbf{Skip})$$

end

Managed Thread

```
1  public class Identifier extends ManagedThread
2  {
3     FieldDeclaration_1
4     . . .
5     FieldDeclaration_n
6
7     ConstructorDeclaration
8
9     run
10
11    AppMeth_1
12    . . .
13    AppMeth_n
14 }
```

$\textbf{process}\,[\![\,PName\,]\!]\,App \mathrel{\widehat{=}} [\![\,[\![\,ConstructorDeclaration\,]\!]_{Method}\,]\!]_{Parameters}\,\textbf{begin}$

State ──────────────────────────────────
$this : \mathrm{ref}\,[\![\,Identifier\,]\!]_{name}\,Class$
──

$\textbf{state}\,State$

Init ──────────────────────────────────
$State'$
$this := \textbf{new}\,[\![\,Identifier\,]\!]_{name}\,Class()$
──

$Run \mathrel{\widehat{=}}$
$$\begin{pmatrix} runCall\,.\,IdOf(PName) \longrightarrow \\ [\![\,[\![\,run\,]\!]_{Method}\,]\!]_{MethBody}; \\ runRet\,.\,IfOf(PName) \longrightarrow \\ \textbf{Skip} \end{pmatrix}$$

$[\![\,AppMeth\_1\,]\!]_{Method}$

. . .

$[\![\,AppMeth\_n\,]\!]_{Method}$

$Methods \mathrel{\widehat{=}}$
$$\begin{pmatrix} Run \\ \Box \\ MethName(AppMeth\_1) \\ \Box \\ MethName(AppMeth\_n) \\ \cdots \end{pmatrix} ;\ Methods$$

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

$\textbf{end}$

Data Class

class $[\![PName]\!]_{name}$ $Class \mathbin{\widehat{=}}$ **begin**

state $State$
$[\![VarName]\!]_{name} : [\![VarType]\!]_{type}$

state $State$

initial $Init$
$State'$
$[\![VarName]\!]'_{name} = [\![VarInit]\!]_{expression}$

$[\![DataMeth1]\!]_{dataMeth}$
$[\![DataMeth2]\!]_{dataMeth}$
. . .

- **Skip**

end