Translation Rules

BNF Encodings

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

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

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

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

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

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

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

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

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

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

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

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

$$SCDeclaration ::= chanName\langle\langle \mathrm{seq}\, N \rangle\rangle \mid chanNameWithType\langle\langle \mathrm{seq}\, N \times Expression \rangle\rangle \mid scSe\langle\langle SchemaExp \rangle\rangle$$

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

$$ChannelDefinition == CDeclaration$$

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

$$CircusProgram == \mathrm{seq}\, CircusParagraph$$

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

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

$Run == MethodBody$
$ManagedThreadClassBody == Run \times \text{seq}\ ClassBodyDeclaration$
$ManagedThread == Identifier \times ManagedThreadClassBody$

$HandleAsyncEvent == MethodBody$
$HandleAsyncLongEvent == Long \times MethodBody$
$EventHandlerClassBody == HandleAsyncEvent \times \text{seq}\ ClassBodyDeclaration$
$OneShotEventHandler == Identifier \times EventHandlerClassBody$
$LongEventHandlerClassBody == HandleAsyncLongEvent \times \text{seq}\ ClassBodyDeclaration$
$AperiodicEventHandler ::= apehType\langle\langle Identifier \times EventHandlerClassBody\rangle\rangle \mid$
$\quad aplehType\langle\langle Identifier \times LongEventHandlerClassBody\rangle\rangle$
$PeriodicEventHandler == Identifier \times EventHandlerClassBody$

$EventHandler ::= pehDecl\langle\langle PeriodicEventHandler\rangle\rangle$
$\quad \mid apehDecl\langle\langle AperiodicEventHandler\rangle\rangle$
$\quad \mid osehDecl\langle\langle OneShotEventHandler\rangle\rangle$

$GetNextMission == MethodBody$
$MissionSequencerClassBody == GetNextMission \times \text{seq}\ ClassBodyDeclaration$
$MissionSequencer == Identifier \times MissionSequencerClassBody$

$NestedMissionSequencer == MissionSequencer$

$SchedulableObject ::= handler\langle\langle EventHandler\rangle\rangle$
$\quad \mid mt\langle\langle ManagedThread\rangle\rangle$
$\quad \mid nms\langle\langle NestedMissionSequencer\rangle\rangle$

$Cleanup == MethodBody$
$Initialize == MethodBody$
$MissionClassBody == Initialize \times Cleanup \times \text{seq}\ ClassBodyDeclaration$
$Mission == Identifier \times MissionClassBody$

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

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

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

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

$$ProgSafelet : SCJProgram \to Safelet$$

$$\forall\, s : SCJProgram \bullet ProgSafelet(s) = s.1$$

$$ProgTLMS : SCJProgram \to TopLevelMissionSequencer$$

$$\forall\, s : SCJProgram \bullet ProgTLMS(s) = s.2$$

$$ProgTiers : SCJProgram \to \text{seq}\, Tier$$

$$\forall\, s : SCJProgram \bullet ProgTiers(s) = s.3$$

Framework

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

[*ID*]

[*Type*]

$$SafeletFWName : N$$
$$TopLevelMissionSequencerFWNMame : N$$

$$controlTierSync : CSExpression$$
$$Tier0 : N$$
$$MissionIds : \text{seq } CircusParagraph$$
$$SchedulableIds : \text{seq } CircusParagraph$$
$$ThreadIds : \text{seq } CircusParagraph$$
$$ObjectIds : \text{seq } CircusParagraph$$

$$ServicesChan : \text{seq } CircusParagraph$$
$$GlobalTypes : \text{seq } CircusParagraph$$
$$JTime : \text{seq } CircusParagraph$$
$$PrimitiveTypes : \text{seq } CircusParagraph$$
$$Priority : \text{seq } CircusParagraph$$
$$PriorityQueue : \text{seq } CircusParagraph$$
$$FrameworkChan : \text{seq } CircusParagraph$$
$$MissionId : \text{seq } CircusParagraph$$
$$SchedulableId : \text{seq } CircusParagraph$$

$$ObjectFW : CircusParagraph$$
$$ObjectChan : \text{seq } CircusParagraph$$
$$ObjectFWChan : \text{seq } CircusParagraph$$
$$ObjectMethChan : \text{seq } CircusParagraph$$
$$ThreadFW : CircusParagraph$$
$$ThreadChan : \text{seq } CircusParagraph$$
$$ThreadFWChan : \text{seq } CircusParagraph$$
$$ThreadMethChan : \text{seq } CircusParagraph$$

$$SafeletFW : CircusParagraph$$
$$SafeletFWChan : \text{seq } CircusParagraph$$
$$SafeletChan : \text{seq } CircusParagraph$$
$$SafeletMethChan : \text{seq } CircusParagraph$$

$TopLevelMissionSequencerFW : CircusParagraph$
$TopLevelMissionSequencerChan : \text{seq } CircusParagraph$
$TopLevelMissionSequencerFWChan : \text{seq } CircusParagraph$

$MissionSequencerChan : \text{seq } CircusParagraph$
$MissionSequencerFWChan : \text{seq } CircusParagraph$
$MissionSequencerMethChan : \text{seq } CircusParagraph$

$MissionFW : CircusParagraph$
$MissionChan : \text{seq } CircusParagraph$
$MissionFWChan : \text{seq } CircusParagraph$
$MissionMethChan : \text{seq } CircusParagraph$

$SchedulableChan : \text{seq } CircusParagraph$
$SchedulableMethChan : \text{seq } CircusParagraph$
$SchedulableFWChan : \text{seq } CircusParagraph$
$HandlerChan : \text{seq } CircusParagraph$
$HandlerFWChan : \text{seq } CircusParagraph$
$HandlerMethChan : \text{seq } CircusParagraph$

$PeriodicEventHandlerChan : \text{seq } CircusParagraph$
$PeriodicEventHandlerFW : CircusParagraph$
$PeriodicEventHandlerFWChan : \text{seq } CircusParagraph$
$PeriodicParameters : \text{seq } CircusParagraph$

$AperiodicEventHandlerChan : \text{seq } CircusParagraph$
$AperiodicEventHandlerFW : CircusParagraph$
$AperiodicLongEventHandlerMethChan : \text{seq } CircusParagraph$
$AperiodicParameters : \text{seq } CircusParagraph$

$OneShotEventHandlerChan : \text{seq } CircusParagraph$
$OneShotEventHandlerFW : CircusParagraph$
$OneShotEventHandlerFWChan : \text{seq } CircusParagraph$
$OneShotEventHandlerMethChan : \text{seq } CircusParagraph$

$SchedulableMissionSequencerFW : CircusParagraph$
$SchedulableMissionSequencerChan : \text{seq } CircusParagraph$
$SchedulableMissionSequencerFWChan : \text{seq } CircusParagraph$

$ManagedThreadFW : CircusParagraph$
$ManagedThreadChan : \text{seq } CircusParagraph$
$ManagedThreadFWChan : \text{seq } CircusParagraph$
$ManagedThreadMethChan : \text{seq } CircusParagraph$

$framework : CircusProgram$

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

Build Phase

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

$$
\begin{array}{|l}
TranslatablePrograms : \mathbb{P}\, SCJProgram \\
\hline
TranslatablePrograms \subset SCJProgram \\
\forall\, s : SCJProgram \\
\quad \bullet\ s \in TranslatablePrograms \\
\qquad \Leftrightarrow ProgTLMS(s) \neq NoSequencer \\
\qquad \land\ ProgTiers(s) \neq \langle\rangle \\
\qquad \land\ \forall\, c : Cluster \\
\qquad\quad \mid \exists\, t : Tier;\ tiers : \mathrm{seq}\ Tier \\
\qquad\qquad \bullet\ tiers = ProgTiers(s) \\
\qquad\qquad \land\ t \in ran\ tiers \\
\qquad\qquad \land\ c \in ran\ t \\
\qquad \bullet\ c.2 \neq \varnothing
\end{array}
$$

$$
\begin{array}{|l}
\hline
AppEnv \\
Name : N \\
Parameters : \mathrm{seq}\ Expression \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\hline
ClusterAppEnv \\
Mission : AppEnv \\
Schedulables : \mathbb{F}\, AppEnv \\
\hline
Schedulables \neq \varnothing \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\hline
TierAppEnv \\
Clusters : \mathrm{seq}\ ClusterAppEnv \\
\hline
Clusters \neq \langle\rangle \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
\hline
AppProcEnv \\
Safelet : AppEnv \\
TopLevelMS : AppEnv \\
Tiers : \mathrm{seq}\ TierAppEnv \\
\hline
Tiers \neq \langle\rangle \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
GetSafeletAppEnv : AppProcEnv \rightarrow AppEnv \\
\hline
\forall\, a : AppProcEnv \bullet \\
\quad GetSafeletAppEnv(a) = a.Safelet
\end{array}
$$

$$GetTLMSAppEnv : AppProcEnv \to AppEnv$$

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

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

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

$$IDof : Identifier \nrightarrow N$$

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

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

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

$$BuildClusterAppEnv : Cluster \nrightarrow ClusterAppEnv$$

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

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

$$BuildTierAppEnv : Tier \pfun TierAppEnv$$

$$\begin{array}{l}
\forall\, tier : dom\ BuildTierAppEnv \\
\quad \mid tier \neq \langle\rangle \\
\quad \bullet\ BuildTierAppEnv(tier) = \langle\!|\, Clusters == BuildClusterAppEnvs(tier) \,|\!\rangle
\end{array}$$

$$BuildTiersAppEnv : \mathrm{seq}\ Tier \pfun \mathrm{seq}\ TierAppEnv$$

$$\begin{array}{l}
\forall\, tiers : dom\ BuildTiersAppEnv \\
\quad \mid tiers \neq \langle\rangle \\
\quad \bullet\ \#\, tiers = 1 \Rightarrow BuildTiersAppEnv(tiers) = \langle BuildTierAppEnv(head\ tiers)\rangle \\
\quad \land\ \#\, tiers \geq 1 \Rightarrow BuildTiersAppEnv(tiers) = \\
\qquad \langle BuildTierAppEnv(head\ tiers)\rangle \frown BuildTiersAppEnv(tail\ tiers)
\end{array}$$

$$BuildAppProcEnv : SCJProgram \pfun AppProcEnv$$

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

LockingEnv

$$\begin{array}{l}
Threads : \mathrm{seq}(ThreadIds \times Priority) \\
Objects : \mathrm{seq}\ ObjectIds \\
\hline
Threads \neq \langle\rangle \\
Objects \neq \langle\rangle
\end{array}$$

$$BuildLockEnv : SCJProgram \pfun LockingEnv$$

$$\begin{array}{l}
\forall\, scjProg : dom\ BuildLockEnv \\
\quad \mid scjProg \in TranslatablePrograms \\
\quad \bullet\ \exists\, lockEnv : LockingEnv \\
\qquad \bullet\ BuildLockEnv(scjProg) = lockEnv
\end{array}$$

ClusterEnv

$Mission : Identifier$
$NestedMissionSequencers : \mathbb{F}\ Identifier$
$ManagedThreads : \mathbb{F}\ Identifier$
$PeriodicEventHandlers : \mathbb{F}\ Identifier$
$AperiodicEventHandlers : \mathbb{F}\ Identifier$
$OneShotEventHandlers : \mathbb{F}\ Identifier$

$disjoint\langle NestedMissionSequencers, ManagedThreads, PeriodicEventHandlers,$
$\quad AperiodicEventHandlers, OneShotEventHandlers\rangle$
$\quad \bigcup\{NestedMissionSequencers, ManagedThreads, PeriodicEventHandlers,$
$\quad AperiodicEventHandlers, OneShotEventHandlers\} \neq \varnothing$

TierEnv

$Clusters : \text{seq}\ ClusterEnv$

$Clusters \neq \langle\rangle$

FWEnv

$TopLevelMS : Identifier$
$Tiers : \text{seq}\ TierEnv$

$Tiers \neq \langle\rangle$

$GetTierFWEnvs : FWEnv \rightarrow \text{seq}\ TierEnv$

$\forall\ env : FWEnv$
$\quad \bullet\ GetTierFWEnvs(env) = env.Tiers$

$GetIdentifiers : \mathbb{F}\ SchedulableObject \nrightarrow \mathbb{F}\ Identifier$

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

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

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

$$
\begin{array}{l}
BuildClusterEnvs : \mathrm{seq}\, Cluster \pfun \mathrm{seq}\, ClusterEnv \\
\hline
\forall\, c : dom\, BuildClusterEnvs \\
\quad \mid c \neq \langle\rangle \land \forall\, s : \mathrm{seq}\, Cluster \bullet s \neq \langle\rangle \\
\quad \bullet \#\, c = 1 \Rightarrow BuildClusterEnvs(c) = \langle BuildClusterEnv(head\ c)\rangle \\
\quad \land\ \#\, c \geq 1 \Rightarrow BuildClusterEnvs(c) = \langle BuildClusterEnv(head\ c)\rangle \frown BuildClusterEnvs(tail\ c)
\end{array}
$$

$$
\begin{array}{l}
BuildTierEnv : Tier \pfun TierEnv \\
\hline
\forall\, tier : \mathrm{seq}\, Cluster \\
\quad \bullet BuildTierEnv(tier) = \langle\!| Clusters == BuildClusterEnvs(tier) |\!\rangle
\end{array}
$$

$$BuildTierEnvs : \text{seq}\ Tier \pfun \text{seq}\ TierEnv$$

$$\forall\, tiers : \text{seq}\ Tier \bullet BuildTierEnvs(tiers) = \langle BuildTierEnv(head\ tiers)\rangle \frown BuildTierEnvs(tail\ tiers)$$

$$BuildFWEnv : SCJProgram \pfun FWEnv$$

$$\begin{array}{l}\forall\, scjProg : dom\ BuildFWEnv \\ \quad \mid scjProg \in TranslatablePrograms \\ \quad \bullet \exists\, tlms : MissionSequencer;\ tlmsID : Identifier;\ tlmsBody : MissionSequencerClassBody; \\ \qquad tiers : \text{seq}\ Tier \mid \\ \qquad\quad scjProg.2 \neq NoSequencer \Rightarrow tlms = (tlmsID, tlmsBody) \\ \qquad\quad \land\ tiers = scjProg.3 \bullet \\ \qquad\quad BuildFWEnv(scjProg) = \\ \qquad\qquad \langle\!|\, TopLevelMS == tlms.1, Tiers == BuildTierEnvs(tiers) \,|\!\rangle\end{array}$$

BinderMethodEnv

$$\begin{array}{l}MethodName : N \\ Locs : \mathbb{F}\ N \\ Callers : \mathbb{F}\ N \\ ReturnType : Type \\ Params : \text{seq}\ Type \\ Synchrinised : \mathbb{B} \\ LocParam : N \\ LocType : Type \\ CallerType : Type\end{array}$$

MCBEnv

$$BinderMethods : \text{seq}\ BinderMethodEnv$$

$$BinderMethods \neq \langle\rangle$$

$$BuildBinderMethodName : N \to N$$

$$GetSFMethods : Safelet \to \text{seq}\ ClassBodyDeclaration$$

$$\forall\, sf : Safelet \bullet GetSFMethods(sf) = sf.2.4$$

$$GetTLMSMethods : MissionSequencer \to \text{seq}\ ClassBodyDeclaration$$

$$\forall\, tlms : MissionSequencer \bullet GetTLMSMethods(tlms) = tlms.2.2$$

$$BuildSFMCBEnv : \text{seq}\ ClassBodyDeclaration \to \text{seq}\ BinderMethodEnv$$

$$\begin{array}{l}\forall\, body : dom\ BuildSFMCBEnv \\ \quad \mid body \neq \langle\rangle \land \forall\, b : \text{seq}\ ClassBodyDeclaration \bullet b \neq \langle\rangle \\ \quad \bullet BuildSFMCBEnv(body) = \langle\rangle\end{array}$$

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

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

$$BuildMCBEnv : SCJProgram \nrightarrow MCBEnv$$

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

Generate Phase

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

$$procNameOf : Process \rightarrow N$$

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

$$TierSync : TierEnv \rightarrow CSExpression$$

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

$$GetMissionID : ClusterEnv \pfun N$$

$$GenerateTiersFWProc : ClusterEnv \pfun Process$$

$$GenerateClusterFWProcs : \operatorname{seq} ClusterEnv \pfun Process$$

$$\begin{aligned}
&\forall clusters : dom\ GenerateClusterFWProcs \mid clusters \neq \langle\rangle \bullet \\
&\quad \#clusters = 1 \Rightarrow GenerateClusterFWProcs(clusters) = \\
&\qquad procPar(\\
&\qquad\quad procName(GetMissionID(head\ clusters)), \\
&\qquad\quad MissionSync, \\
&\qquad\quad GenerateTiersFWProc(head\ clusters) \\
&\qquad) \\
&\quad \land \#clusters \geq 1 \Rightarrow GenerateClusterFWProcs(clusters) = \\
&\qquad procPar(\\
&\qquad\quad procPar(\\
&\qquad\qquad procName(GetMissionID(head\ clusters)), \\
&\qquad\qquad MissionSync, \\
&\qquad\qquad GenerateTiersFWProc(head\ clusters)), \\
&\qquad\quad SchedulablesSync, \\
&\qquad\quad GenerateClusterFWProcs(tail\ clusters) \\
&\qquad)
\end{aligned}$$

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

$$\begin{array}{l}
\forall\, tiers : \text{seq}\ TierEnv \\
\quad \mid tiers \neq \langle\rangle \\
\quad \bullet\ \#\, tiers = 1 \Rightarrow \\
\qquad GenerateTierFWProcs(tiers) = \langle GenerateClusterFWProcs((head\ tiers).Clusters)\rangle \\
\quad \land\ \#\, tiers \geq 1 \Rightarrow \\
\qquad GenerateTierFWProcs(tiers) = \\
\qquad\quad \langle GenerateClusterFWProcs((head\ tiers).Clusters)\rangle \\
\qquad\quad \frown GenerateTierFWProcs(tail\ tiers)
\end{array}$$

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

$$ControlTier : N$$
$$TopLevelMissionSequencerFWName : N$$

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

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

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

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

$$GenerateAppProc : AppProcEnv \pfun Process$$

$$\begin{array}{l}
\forall\, appProcEnv : AppProcEnv \\
\quad \bullet\ \exists\, sfAppEnv : AppEnv;\ tlmsAppEnv : AppEnv;\ tiersAppEnvs : \text{seq}\ TierAppEnv \\
\qquad \mid sfAppEnv = GetSafeletAppEnv(appProcEnv) \\
\qquad \land\ tlmsAppEnv = GetTLMSAppEnv(appProcEnv) \\
\qquad \land\ tiersAppEnvs = GetTiersAppEnv(appProcEnv) \\
\qquad \bullet\ GenerateAppProc(appProcEnv) = \\
\qquad\quad procInter(\\
\qquad\qquad procInter(\\
\qquad\qquad\quad procInstP(procName(sfAppEnv.Name), sfAppEnv.Parameters), \\
\qquad\qquad\quad procInstP(procName(tlmsAppEnv.Name), tlmsAppEnv.Parameters) \\
\qquad\qquad), \\
\qquad\quad GenerateAppTierProcs(tiersAppEnvs) \\
\qquad)
\end{array}$$

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

$BinderCallChan : N \pfun \operator{seq} N$

$NaturalCallChan : N \pfun \operator{seq} N$

$NaturalRetChan : N \pfun \operator{seq} N$

$BindeRetChan : N \pfun \operator{seq} N$

$MCBParams : \operator{seq} Type \pfun Expression$

$GenerateMCBChan : BinderMethodEnv \fun CircusParagraph$

$$\begin{array}{l}\forall\, bme : BinderMethodEnv \\ \quad \bullet\ GenerateMCBChan(bme) = chanDef(\\ \qquad multiDecl(chanNameWithType(NaturalCallChan(bme.MethodName), \\ \qquad\quad MCBParams(bme.Params)), \\ \qquad scDecl(chanNameWithType(NaturalRetChan(bme.MethodName), \\ \qquad\quad MCBParams(bme.Params)))) \\ \quad)\end{array}$$

$MethodCallBinderSync : N$
$GenerateMethodCallBinderSync : \operator{seq} BinderMethodEnv \pfun CircusParagraph$

$GenerateMCBChans : \operator{seq} BinderMethodEnv \fun \operator{seq} CircusParagraph$

$$\begin{array}{l}\forall\, bEnvs : \operator{seq} BinderMethodEnv \\ \quad \mid bEnvs \neq \langle\rangle \\ \quad \bullet\ \# bEnvs = 1 \Rightarrow \\ \qquad GenerateMCBChans(bEnvs) = \langle GenerateMCBChan(head\ bEnvs)\rangle \\ \quad \land\ \# bEnvs \geq 1 \Rightarrow \\ \qquad GenerateMCBChans(bEnvs) = \langle GenerateMCBChan(head\ bEnvs)\rangle \\ \qquad\quad \frown GenerateMCBChans(tail\ bEnvs)\end{array}$$

$BinderCallComm : N \pfun N$

$NaturalCallComm : N \pfun N$

$NaturalRetComm : N \pfun N$

$BindeRetComm : N \pfun N$

$GenerateMCBName : N \pfun N$

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

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

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

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

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

$GenerateMCBAction : BinderMethodEnv \to PParagraph$

$$
\begin{aligned}
&\forall\, bme : BinderMethodEnv \\
&\quad \bullet\ GenerateMCBAction(bme) = actDef(GenerateMCBName(bme.MethodName), \\
&\qquad prefixExp((BinderCallComm(bme.MethodName), \\
&\qquad\qquad BinderCallParams(bme.Params)), \\
&\qquad\quad prefixExp((NaturalCallComm(bme.MethodName), \\
&\qquad\qquad\quad BinderCallParams(bme.Params)), \\
&\qquad\qquad prefixExp((NaturalRetComm(bme.MethodName), \\
&\qquad\qquad\qquad BinderCallParams(bme.Params)), \\
&\qquad\qquad\quad prefixExp((BindeRetComm(bme.MethodName), \\
&\qquad\qquad\qquad\quad BinderCallParams(bme.Params)), \\
&\qquad\qquad\qquad\quad actName(GenerateMCBName(bme.MethodName)) \\
&\qquad\qquad\quad) \\
&\qquad\qquad) \\
&\qquad\quad) \\
&\qquad) \\
&\quad)
\end{aligned}
$$

$$GenerateMCBActions : \text{seq}\, BinderMethodEnv \pfun \text{seq}\, PParagraph$$

$$\begin{array}{l}
\forall\, bEnvs : \text{seq}\, BinderMethodEnv \\
\quad \mid bEnvs \neq \langle\rangle \\
\quad \bullet\ \#\, bEnvs = 1 \Rightarrow \\
\qquad GenerateMCBActions(bEnvs) = \langle GenerateMCBAction(head\ bEnvs)\rangle \\
\quad \land\ \#\, bEnvs \geq 1 \Rightarrow \\
\qquad GenerateMCBActions(bEnvs) = \langle GenerateMCBAction(head\ bEnvs)\rangle \\
\qquad\quad \frown GenerateMCBActions(tail\ bEnvs)
\end{array}$$

$$GenerateMCBProc : \text{seq}\, BinderMethodEnv \pfun CircusParagraph$$

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

$$GenerateMCBModel : MCBEnv \pfun \text{seq}\, CircusParagraph$$

$$\begin{array}{l}
\forall\, bEnv : MCBEnv \\
\quad \bullet\ GenerateMCBModel(bEnv) = GenerateMCBChans(bEnv.BinderMethods) \frown \\
\qquad \langle GenerateMethodCallBinderSync(bEnv.BinderMethods), GenerateMCBProc(bEnv.Bin
\end{array}$$

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

$$GenerateObjectProc : \text{seq}\, ObjectIds \pfun Process$$

$$GenerateLockModel : LockingEnv \pfun \text{seq}\, CircusParagraph$$

$$\begin{array}{l}
\forall\, lEnv : dom\ GenerateLockModel \\
\quad \mid lEnv.Threads \neq \langle\rangle \land lEnv.Objects \neq \langle\rangle \\
\quad \bullet\ GenerateLockModel(lEnv) = \\
\qquad \langle \\
\qquad\quad procDef(pd(Locking, procPar(procName(Threads), \\
\qquad\qquad ThreadSync, \\
\qquad\qquad procName(Objects))) \\
\qquad\quad), \\
\qquad\quad procDef(pd(Threads, GenerateThreadProc(lEnv.Threads))), \\
\qquad\quad procDef(pd(Objects, GenerateObjectProc(lEnv.Objects))) \\
\qquad \rangle
\end{array}$$

Translate SCJ Program

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

$$ProcessID : N \pfun ID$$

$$TransClasses : SCJProgram \pfun CircusProgram$$

$$FWName : N$$
$$AppName : N$$
$$MCBName : N$$
$$LockName : N$$
$$ProgName : Identifier \pfun N$$

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

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

Low Level

- $Method : MethodDeclaration \pfun (Name, Params, ReturnType, Body)$: translates an active application method into a *Circus* action
- $DataMethod : MethodDeclaration \pfun$: translates data methods into an *OhCircus* method
- $MethodBody : Block \pfun \seq CircExpression$: translates a Java block, for example a method body
- $Registers : Block \pfun \seq Name$: extracts the Names of the schedulables registered in a Java block
- $Returns : Block \pfun \seq Name$: extracts the Names of the variables retuned in a Java block
- $Variable : (Name, Type, InitExpression) \pfun (CircName, CircType, CircExpression)$: translates a variable
- $Parameters : (Name, Params, ReturnType, Body) \pfun \seq CircParam$: 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 }
```

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

State

$this : \text{ref } [\![Identifier]\!]_{name}\ Class$

state *State*

Init

$State'$

$this := \textbf{new } [\![Identifier]\!]_{name}\ Class()$

$$InitializeApplication \mathrel{\widehat=} \left(\begin{array}{l} initializeApplicationCall \longrightarrow \\ [\![[\![InitializeApplication]\!]_{Method}]\!]_{MethBody} \\ initializeApplicationRet \longrightarrow \\ \textbf{Skip} \end{array} \right)$$

$$GetSequencer \mathrel{\widehat=} \left(\begin{array}{l} getSequencerCall \longrightarrow \\ getSequencerRet\,!\ [\![GetSequencer]\!]_{Returns} \longrightarrow \\ \textbf{Skip} \end{array} \right)$$

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

...

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

$$
\begin{array}{l}
Methods \mathrel{\widehat{=}} \\
\left(\begin{array}{l}
GetSequencer \\
\Box \\
InitializeApplication \\
\Box \\
MethName(AppMeth\_1) \\
\Box \\
\ldots \\
MethName(AppMeth\_n) \\
\ldots
\end{array}\right) \; ; \; Methods
\end{array}
$$

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

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 }
```

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

State

$$this : \text{ref}\ [\![Identifier]\!]_{name}\ Class$$

$$\textbf{state}\ State$$

Init

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

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

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

$$\ldots$$

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

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

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

$$\textbf{end}$$

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 }
```

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

State

$this : \text{ref}\ [\![Identifier]\!]_{name}\ Class$

state *State*

Init

$State'$

$this := \textbf{new}\ [\![Identifier]\!]_{name}\ Class()$

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

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

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

...

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

$$Methods \mathrel{\widehat=} \begin{pmatrix} InitializePhase \\ \Box \\ CleanupPhase \\ \Box \\ MethName(AppMeth\_1) \\ \Box \\ MethName(AppMeth\_n) \\ \ldots \end{pmatrix} ;\ Methods$$

• $(Init\ ;\ Methods)\ \ \triangle(end\_mission\_app\ .\ IdOf(Identifier) \longrightarrow \textbf{Skip}$

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 }
```

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

State

$$this : \text{ref}\ [\![Identifier]\!]_{name}\ Class$$

$$\textbf{state}\ State$$

Init

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

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

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

$$\ldots$$

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

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

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

$$\textbf{end}$$

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

$$this : \text{ref}\ [\![Identifier]\!]_{name}\ Class$$

$$\textbf{state}\ State$$

Init

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

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

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

$$\ldots$$

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

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

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

$$\textbf{end}$$

Data Class

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

state *State*

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

state *State*

initial *Init*

$State'$

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

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

- **Skip**

end