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*

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

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

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

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

$NestedMissionSequencer == MissionSequencer$

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

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

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

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

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

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

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*

$$AppEnv == N \times \text{seq}\, Expression$$

$$ClusterAppEnv == AppEnv \times \text{seq}\, AppEnv$$

$$TierAppEnv == \text{seq}\, ClusterAppEnv$$

$$TiersAppEnv == \text{seq}\, TierAppEnv$$

$$AppProcEnv == AppEnv \times AppEnv \times TiersAppEnv$$

$$GetSafeletAppEnv : AppProcEnv \rightarrow AppEnv$$

$$\forall a : AppProcEnv \bullet GetSafeletAppEnv(a) = a.1$$

$$GetTLMSAppEnv : AppProcEnv \rightarrow AppEnv$$

$$\forall a : AppProcEnv \bullet GetTLMSAppEnv(a) = a.2$$

$$GetTiersAppEnv : AppProcEnv \rightarrow TiersAppEnv$$

$$\forall a : AppProcEnv \bullet GetTiersAppEnv(a) = a.3$$

$$IDof : Identifier \pfun N$$

$$ParamsOf : \text{seq}\, ClassBodyDeclaration \pfun \text{seq}\, Expression$$

$$BuildSOAppEnv : \operatorname{seq} SchedulableObject \pfun \operatorname{seq} AppEnv$$

$$\begin{array}{l}
\forall\, scheds : \operatorname{seq} SchedulableObject \\
\quad \mid scheds \neq \langle\rangle \\
\quad \bullet\ \exists\, manT : ManagedThread;\ nestMS : NestedMissionSequencer;\ eh : EventHandler \\
\qquad perEH : PeriodicEventHandler;\ oneEH : OneShotEventHandler; \\
\qquad apehShort : Identifier \times EventHandlerClassBody; \\
\qquad apehLong : Identifier \times LongEventHandlerClassBody \\
\qquad \bullet\ head\ scheds = mt(manT) \Rightarrow \\
\qquad\quad BuildSOAppEnv(scheds) = \langle (IDof(manT.1), ParamsOf(manT.2.2)) \rangle \\
\qquad\qquad \frown BuildSOAppEnv(tail\ scheds) \\
\qquad \land\ head\ scheds = nms(nestMS) \Rightarrow \\
\qquad\quad BuildSOAppEnv(scheds) = \langle (IDof(nestMS.1), ParamsOf(nestMS.2.2)) \rangle \\
\qquad\qquad \frown BuildSOAppEnv(tail\ scheds) \\
\qquad \land\ head\ scheds = handler(pehDecl(perEH)) \Rightarrow \\
\qquad\quad BuildSOAppEnv(scheds) = \langle (IDof(perEH.1), ParamsOf(perEH.2.2)) \rangle \\
\qquad\qquad \frown BuildSOAppEnv(tail\ scheds) \\
\qquad \land\ head\ scheds = handler(osehDecl(oneEH)) \Rightarrow \\
\qquad\quad BuildSOAppEnv(scheds) = \langle (IDof(oneEH.1), ParamsOf(oneEH.2.2)) \rangle \\
\qquad\qquad \frown BuildSOAppEnv(tail\ scheds) \\
\qquad \land\ head\ scheds = handler(apehDecl(apehType(apehShort))) \Rightarrow \\
\qquad\quad BuildSOAppEnv(scheds) = \langle (IDof(apehShort.1), ParamsOf(apehShort.2.2)) \rangle \\
\qquad\qquad \frown BuildSOAppEnv(tail\ scheds) \\
\qquad \land\ head\ scheds = handler(apehDecl(aplehType(apehLong))) \Rightarrow \\
\qquad\quad BuildSOAppEnv(scheds) = \langle (IDof(apehLong.1), ParamsOf(apehLong.2.2)) \rangle \\
\qquad\qquad \frown BuildSOAppEnv(tail\ scheds)
\end{array}$$

$$BuildClusterAppEnv : Cluster \pfun AppEnv \times \operatorname{seq} AppEnv$$

$$\begin{array}{l}
\forall\, c : Cluster \\
\quad \mid c.2 \neq \langle\rangle \\
\quad \bullet\ \exists\, m : Mission;\ seqSO : \operatorname{seq} SchedulableObject \\
\qquad \mid c = (m, seqSO) \\
\qquad \bullet\ BuildClusterAppEnv(c) = \\
\qquad\quad ((IDof(m.1), ParamsOf(m.2.3)), BuildSOAppEnv(seqSO))
\end{array}$$

$$BuildTierAppEnv : Tier \pfun TierAppEnv$$

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

$$BuildTiersAppEnv : \operatorname{seq} Tier \pfun TiersAppEnv$$

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

$$BuildAppProcEnv : SCJProgram \pfun AppProcEnv$$

$$\begin{array}{l}
\forall\, scjProg : SCJProgram \bullet \\
\quad \exists\, safelet : Safelet;\ tiers : \mathrm{seq}\ Tier;\ ms : MissionSequencer \\
\quad\quad \mid scjProg = (safelet, tlms(ms), tiers) \bullet \\
\quad\quad\quad \exists\, sfEnv : AppEnv;\ tlmsEnv : AppEnv; \\
\quad\quad\quad\quad tiersEnv : TiersAppEnv \bullet \\
\quad\quad\quad\quad\quad sfEnv = (IDof(safelet.1), ParamsOf(safelet.2.4)) \land \\
\quad\quad\quad\quad\quad tlmsEnv = (IDof(ms.1), ParamsOf(ms.2.2)) \land \\
\quad\quad\quad\quad\quad tiersEnv = BuildTiersAppEnv(tiers) \land \\
\quad\quad\quad\quad\quad BuildAppProcEnv(scjProg) = (sfEnv, tlmsEnv, tiersEnv)
\end{array}$$

$$LockingEnv == \mathrm{seq}(ThreadIds \times Priority) \times \mathrm{seq}\ ObjectIds$$

$$BuildLockEnv : SCJProgram \pfun LockingEnv$$

$$GetIdentifiers : \mathrm{seq}\ SchedulableObject \pfun \mathrm{seq}\ Identifier$$

$$\begin{array}{l}
\forall\, scheds : \mathrm{seq}\ SchedulableObject \\
\quad \mid scheds \neq \langle\rangle \\
\quad \bullet\ \exists\, ident : Identifier;\ manT : ManagedThread; \\
\quad\quad nestMS : NestedMissionSequencer;\ eh : EventHandler \\
\quad\quad perEH : PeriodicEventHandler;\ oneEH : OneShotEventHandler; \\
\quad\quad apehShort : Identifier \times EventHandlerClassBody; \\
\quad\quad apehLong : Identifier \times LongEventHandlerClassBody \\
\quad\quad \mid head\ scheds = mt(manT) \Rightarrow ident = manT.1 \\
\quad\quad \land\ head\ scheds = nms(nestMS) \Rightarrow ident = nestMS.1 \\
\quad\quad \land\ head\ scheds = handler(pehDecl(perEH)) \Rightarrow ident = perEH.1 \\
\quad\quad \land\ head\ scheds = handler(apehDecl(apehType(apehShort))) \Rightarrow ident = apehShort.1 \\
\quad\quad \land\ head\ scheds = handler(apehDecl(aplehType(apehLong))) \Rightarrow ident = apehLong.1 \\
\quad\quad \land\ head\ scheds = handler(osehDecl(oneEH)) \Rightarrow ident = oneEH.1 \\
\quad\quad\quad \bullet\ \#\, scheds = 1 \Rightarrow GetIdentifiers(scheds) = \langle ident \rangle \\
\quad\quad\quad \land\ \#\, scheds \geq 1 \Rightarrow GetIdentifiers(scheds) = \langle ident \rangle \frown GetIdentifiers(tail\ scheds)
\end{array}$$

$$\begin{array}{l}
BuildSOEnvs : \mathrm{seq}\ SchedulableObject \pfun \\
\quad \mathrm{seq}\ Identifier \times \mathrm{seq}\ Identifier \times \mathrm{seq}\ Identifier \times \\
\quad \mathrm{seq}\ Identifier \times \mathrm{seq}\ Identifier
\end{array}$$

$$\begin{array}{l}
\forall\, s : \mathrm{seq}\ SchedulableObject \\
\quad \mid s \neq \langle\rangle \\
\quad \bullet\ \exists\, sms : \mathrm{seq}\ Identifier;\ pehs : \mathrm{seq}\ Identifier; \\
\quad\quad\quad apehs : \mathrm{seq}\ Identifier;\ osehs : \mathrm{seq}\ Identifier;\ mts : \mathrm{seq}\ Identifier \\
\quad\quad \mid mts = GetIdentifiers(s \rhd \{m : ManagedThread \bullet mt(m)\}) \\
\quad\quad \land\ sms = GetIdentifiers(s \rhd \{n : NestedMissionSequencer \bullet nms(n)\}) \\
\quad\quad \land\ pehs = GetIdentifiers(s \rhd \{p : PeriodicEventHandler \bullet handler(pehDecl(p))\}) \\
\quad\quad \land\ apehs = GetIdentifiers(s \rhd \{a : Identifier \times EventHandlerClassBody \\
\quad\quad\quad\quad \bullet handler(apehDecl(apehType(a)))\}) \\
\quad\quad \land\ apehs = GetIdentifiers(s \rhd \{a : Identifier \times LongEventHandlerClassBody \\
\quad\quad\quad\quad \bullet handler(apehDecl(aplehType(a)))\}) \\
\quad\quad \land\ osehs = GetIdentifiers(s \rhd \{o : OneShotEventHandler \\
\quad\quad\quad\quad \bullet handler(osehDecl(o))\}) \\
\quad \bullet\ BuildSOEnvs(s) = (sms, pehs, apehs, osehs, mts)
\end{array}$$

$$ClusterEnv == Identifier \times \operatorname{seq} Identifier \times \operatorname{seq} Identifier \times \operatorname{seq} Identifier \times \operatorname{seq} Identifier \times \operatorname{seq} Identifier$$

$$TierEnv == \operatorname{seq} ClusterEnv$$

$$FWEnv == Identifier \times \operatorname{seq} TierEnv$$

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

$$\forall env : FWEnv \bullet GetTierFWEnvs(env) = env.2$$

$$BuildClusterEnv : Cluster \pfun ClusterEnv$$

$$\begin{aligned}
&\forall c : Cluster \\
&\quad \mid c.2 \neq \langle\rangle \\
&\quad \bullet \exists missionName : Identifier;\ sms : \operatorname{seq} Identifier;\ pehs : \operatorname{seq} Identifier; \\
&\qquad apehs : \operatorname{seq} Identifier;\ oseh : \operatorname{seq} Identifier;\ mts : \operatorname{seq} Identifier \\
&\quad \mid missionName = c.1.1 \\
&\quad \land (sms, pehs, apehs, oseh, mts) = BuildSOEnvs(c.2) \\
&\quad \bullet BuildClusterEnv(c) = (missionName, sms, pehs, apehs, oseh, mts)
\end{aligned}$$

$$BuildTierEnv : Tier \pfun TierEnv$$

$$\forall t : \operatorname{seq} Cluster \bullet BuildTierEnv(t) = \langle BuildClusterEnv(head\ t)\rangle \frown BuildTierEnv(tail\ t)$$

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

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

$$BuildFWEnv : SCJProgram \pfun FWEnv$$

$$\begin{aligned}
&\forall scjProg : SCJProgram \bullet \\
&\quad \exists tlms : MissionSequencer;\ tlmsID : Identifier;\ tlmsBody : MissionSequencerClassBody; \\
&\qquad tiers : \operatorname{seq} Tier \mid \\
&\qquad\quad scjProg.2 \neq NoSequencer \Rightarrow tlms = (tlmsID, tlmsBody) \\
&\qquad\quad \land\ tiers = scjProg.3 \bullet \\
&\qquad\quad BuildFWEnv(scjProg) = (tlms.1, BuildTierEnvs(tiers))
\end{aligned}$$

$$BinderMethodEnv == N \times \mathbb{F}\, N \times \mathbb{F}\, N \times Type \times \operatorname{seq} Type \times \mathbb{B} \times N \times Type \times Type$$

$$MCBEnv == \text{seq}\ BinderMethodEnv$$

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

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

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

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

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

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

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

$$BuildMCBEnv : SCJProgram \nrightarrow \text{seq}\ BinderMethodEnv$$

$$\begin{array}{l}
\forall\, scjProg : SCJProgram \\
\quad \mid scjProg.2 \neq NoSequencer \\
\quad \bullet \exists\, sf : Safelet;\ tlms : MissionSequencer;\ tiers : \text{seq}\ Tier \\
\qquad \mid sf = scjProg.1 \\
\qquad \land\ tiers = scjProg.3 \\
\qquad \bullet\ BuildMCBEnv(scjProg) = \\
\qquad\quad \langle BuildSFMCBEnv(GetSFMethods(sf)), \\
\qquad\quad BuildTLMSMCBEnv(GetTLMSMethods(tlms))\rangle \\
\qquad\quad \frown BuildTierMCBEnvs(tiers)
\end{array}$$

Generate Phase

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

$$procNameOf : Process \rightarrow N$$

$$TierSync : CSExpression$$
$$ControLTierSync : CSExpression$$
$$MissionSync : CSExpression$$
$$SchedulablesSync : CSExpression$$

$$GetMissionID : ClusterEnv \nrightarrow N$$

$$GetSOIDs : ClusterEnv \nrightarrow \text{seq}\, Identifier \times \text{seq}\, Identifier \times \text{seq}\, Identifier \times \text{seq}\, Identifier \times \text{seq}\, Identifier$$

$$GenerateTiersFWProc : \text{seq}\, Identifier \times \text{seq}\, Identifier \times \text{seq}\, Identifier \times \text{seq}\, Identifier \times \text{seq}\, Identifier \nrightarrow Process$$

$$GenerateClusterFWProcs : TierEnv \nrightarrow Process$$

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

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

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

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

$ControlTier : N$
$ControlTierSync : CSExpression$
$TopLevelMissionSequencerFWName : N$

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

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

$\forall\, env : FWEnv$
$\quad \mid env.2 \neq \langle\rangle$
$\quad \bullet\ \exists\, fwProc : Process;\ controlTierProc : Process;\ tierProcs : \text{seq}\ Process$
$\quad\quad \mid fwProc = procPar(procName(ControlTier), TierSync, GenerateTierFWProc(env.2))$
$\quad\quad \land\ controlTierProc = procPar(procName(SafeletFWName), ControlTierSync,$
$\quad\quad\quad procInstP(procName(TopLevelMissionSequencerFWName), GetParams(env.1)))$
$\quad\quad \land\ tierProcs = GenerateTierFWProcs(env.2)$
$\quad\quad \bullet\ GenerateFWProcs(env) = \langle fwProc\rangle \frown \langle controlTierProc\rangle \frown tierProcs$

$GenerateAppTierProcs : TiersAppEnv \pfun Process$

$GenerateAppProc : AppProcEnv \pfun Process$

$\forall\, appProcEnv : AppProcEnv$
$\quad \bullet\ \exists\, sfAppEnv : AppEnv;\ tlmsAppEnv : AppEnv;\ tiersAppEnvs : TiersAppEnv$
$\quad\quad \mid sfAppEnv = GetSafeletAppEnv(appProcEnv)$
$\quad\quad \land\ tlmsAppEnv = GetTLMSAppEnv(appProcEnv)$
$\quad\quad \land\ tiersAppEnvs = GetTiersAppEnv(appProcEnv)$
$\quad\quad \bullet\ GenerateAppProc(appProcEnv) =$
$\quad\quad\quad procInter($
$\quad\quad\quad\quad procInter($
$\quad\quad\quad\quad\quad procInstP(procName(sfAppEnv.1), sfAppEnv.2),$
$\quad\quad\quad\quad\quad procInstP(procName(tlmsAppEnv.1), tlmsAppEnv.2)$
$\quad\quad\quad\quad),$
$\quad\quad\quad\quad GenerateAppTierProcs(tiersAppEnvs)$
$\quad\quad\quad)$

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

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

$NaturalCallChan : N \pfun \seq N$

$NaturalRetChan : N \pfun \seq N$

$BindeRetChan : N \pfun \seq N$

$MCBParams : \seq Type \pfun Expression$

$GenerateMCBChan : BinderMethodEnv \fun CircusParagraph$

$\forall\, bme : BinderMethodEnv$
$\bullet\ GenerateMCBChan(bme) = chanDef($
$multiDecl(chanNameWithType(BinderCallChan(bme.1), MCBParams(bme.5)),$
$multiDecl(chanNameWithType(NaturalCallChan(bme.1), MCBParams(bme.5)),$
$multiDecl(chanNameWithType(NaturalRetChan(bme.1), MCBParams(bme.5)),$
$scDecl(chanNameWithType(BindeRetChan(bme.1), MCBParams(bme.5))))))$
$)$

$MethodCallBinderSync : N$
$GenerateMethodCallBinderSync : \seq BinderMethodEnv \pfun CircusParagraph$

$GenerateMCBChans : \seq BinderMethodEnv \fun \seq CircusParagraph$

$\forall\, bEnvs : \seq BinderMethodEnv$
$\mid bEnvs \neq \langle\rangle$
$\bullet\ \# bEnvs = 1 \Rimp$
$GenerateMCBChans(bEnvs) = \langle GenerateMCBChan(head\ bEnvs)\rangle$
$\land\ \# bEnvs \geq 1 \Rimp$
$GenerateMCBChans(bEnvs) = \langle GenerateMCBChan(head\ bEnvs)\rangle$
$\frown GenerateMCBChans(tail\ bEnvs)$

$BinderCallComm : N \pfun N$

$NaturalCallComm : N \pfun N$

$NaturalRetComm : N \pfun N$

$BindeRetComm : N \pfun N$

$GenerateMCBName : N \pfun N$

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

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

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

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

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

$GenerateMCBAction : BinderMethodEnv \rightarrow PParagraph$

$$
\begin{array}{l}
\forall\, bme : BinderMethodEnv \\
\quad \bullet\ GenerateMCBAction(bme) = actDef(GenerateMCBName(bme.1), \\
\qquad prefixExp((BinderCallComm(bme.1), BinderCallParams(bme.5)), \\
\qquad\quad prefixExp((NaturalCallComm(bme.1), BinderCallParams(bme.5)), \\
\qquad\qquad prefixExp((NaturalRetComm(bme.1), BinderCallParams(bme.5)), \\
\qquad\qquad\quad prefixExp((BindeRetComm(bme.1), BinderCallParams(bme.5)), \\
\qquad\qquad\qquad actName(GenerateMCBName(bme.1)) \\
\qquad\qquad\quad) \\
\qquad\qquad) \\
\qquad\quad) \\
\qquad) \\
\quad)
\end{array}
$$

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

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

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

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

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

$$\begin{array}{l}
\forall\, bEnvs : \text{seq}\, BinderMethodEnv \\
\quad \mid bEnvs \neq \langle\rangle \\
\quad \bullet\ GenerateMCBModel(bEnvs) = GenerateMCBChans(bEnvs)^\frown \\
\qquad \langle GenerateMethodCallBinderSync(bEnvs), GenerateMCBProc(bEnvs)\rangle
\end{array}$$

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

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

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

$$\begin{array}{l}
\forall\, lEnv : LockingEnv \\
\quad \mid lEnv.1 \neq \langle\rangle \land lEnv.2 \neq \langle\rangle \\
\quad \bullet\ GenerateLockModel(lEnv) = \\
\qquad \langle \\
\qquad\quad procDef(pd(Locking, procPar(procName(Threads), \\
\qquad\qquad\quad ThreadSync, \\
\qquad\qquad\quad procName(Objects))) \\
\qquad\quad), \\
\qquad\quad procDef(pd(Threads, GenerateThreadProc(lEnv.1))), \\
\qquad\quad procDef(pd(Objects, GenerateObjectProc(lEnv.2))) \\
\qquad \rangle
\end{array}$$

Translate SCJ Program

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

$$ProcessID : N \pfun ID$$

$$TransClasses : SCJProgram \pfun CircusProgram$$

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

$$TransSCJProg : Identifier \times SCJProgram \pfun CircusProgram$$

$$\forall\, scjProg : SCJProgram;\ name : Identifier \bullet$$
$$\exists\, app : CircusProgram;$$
$$program : CircusProgram;$$
$$appComms : CSExpression;\ mcbComms : CSExpression;\ lockComms : CSExpression$$
$$fwProcs : \mathrm{seq}\, Process;\ appProc : Process;\ lockModel : \mathrm{seq}\, CircusParagraph;$$
$$mcbModel : \mathrm{seq}\, CircusParagraph \mid$$
$$app = TransClasses(scjProg) \land$$
$$fwProcs = GenerateFWProcs(BuildFWEnv(scjProg)) \land$$
$$appProc = GenerateAppProc(BuildAppProcEnv(scjProg)) \land$$
$$mcbModel = GenerateMCBModel(BuildMCBEnv(scjProg)) \land$$
$$lockModel = GenerateLockModel(BuildLockEnv(scjProg)) \land$$
$$program = \langle procDef(pd(ProgName(name),$$
$$procHide(procPar($$
$$procHide($$
$$procPar($$
$$procName(FWName),$$
$$appComms,$$
$$procHide($$
$$procPar(procName(AppName),$$
$$mcbComms,$$
$$procName(MCBName)),$$
$$mcbComms)),$$
$$appComms),$$
$$lockComms,$$
$$procName(LockName)),$$
$$lockComms)))\rangle \bullet$$
$$TransSCJProg(name, scjProg) =$$
$$framework \frown \langle procDef(pd(FWName, head\, fwProcs))\rangle \frown$$
$$app \frown \langle procDef(pd(AppName, appProc))\rangle \frown$$
$$mcbModel \frown$$
$$lockModel \frown$$
$$program$$

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

$State'$

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

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

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

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

...

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

$$
\begin{array}{l}
Methods \mathrel{\widehat{=}} \\
\left(\begin{array}{l}
GetSequencer \\
\Box \\
InitializeApplication \\
\Box \\
MethName(AppMeth\_1) \\
\Box \\
\ldots \\
MethName(AppMeth\_n) \\
\ldots
\end{array}\right) \;;\; Methods \\
\\
\bullet\ (Init \;;\; Methods) \;\; \triangle (end\_safelet\_app \longrightarrow \mathbf{Skip}) \\
\\
\mathbf{end}
\end{array}
$$

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 }
```

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

State

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

state *State*

Init

$State'$

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

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

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

$\ldots$

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

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

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

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 }
```

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

State

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

state *State*

Init

$State'$

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

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

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

...

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

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

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

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