

Translation Rules

BNF Encodings

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

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

Command ::= *spec*⟨⟨seq *N* × *Predicate* × *Predicate*⟩⟩ | *equals*⟨⟨*N* × seq *Expression*⟩⟩

CParameter ::= *shriek*⟨⟨*N*⟩⟩ | *shriekRestrict*⟨⟨*N* × *Predicate*⟩⟩ | *bang*⟨⟨*Expression*⟩⟩ |
dotParam⟨⟨*Expression*⟩⟩

Communication == *N* × seq *CParameter*

CSEExpression ::= *cs*⟨⟨seq *N*⟩⟩ | *csName*⟨⟨*N*⟩⟩ |
union⟨⟨*CSEExpression* × *CSEExpression*⟩⟩ |
intersect⟨⟨*CSEExpression* × *CSEExpression*⟩⟩ |
subtract⟨⟨*CSEExpression* × *CSEExpression*⟩⟩

Action ::= *actSe*⟨⟨*SchemaExp*⟩⟩ | *com*⟨⟨*Command*⟩⟩ | *skip* | *stop* | *chaos* |
prefixExp⟨⟨*Communication* × *Action*⟩⟩ |
guard⟨⟨*Predicate* × *Action*⟩⟩ | *seqExp*⟨⟨*Action* × *Action*⟩⟩ |
extChoice⟨⟨*Action* × *Action*⟩⟩ | *intChoice*⟨⟨*Action* × *Action*⟩⟩ |
actPar⟨⟨*Action* × *CSEExpression* × *Action*⟩⟩ | *actInter*⟨⟨*Action* × *Action*⟩⟩ |
actHide⟨⟨*Action* × *CSEExpression*⟩⟩ | *mu*⟨⟨*N* × *Action*⟩⟩ | *actParam*⟨⟨*Declaration* × *Action*⟩⟩ |
actInst⟨⟨*Action* × seq *Expression*⟩⟩ | *actName*⟨⟨*N*⟩⟩ | *actInterrupt*⟨⟨*Action* × *Action*⟩⟩

GuardedAction ::= *thenAct*⟨⟨*Predicate* × *Action*⟩⟩ |
thenActComp⟨⟨*Predicate* × *Action* × *GuardedAction*⟩⟩

PParagraph ::= *pPar*⟨⟨*Paragraph*⟩⟩ | *actDef*⟨⟨*N* × *Action*⟩⟩

Process ::= *proc*⟨⟨seq *PParagraph* × *SchemaExp* × seq *PParagraph* × *Action*⟩⟩ | *procName*⟨⟨*N*⟩⟩ |
procSeq⟨⟨*Process* × *Process*⟩⟩ | *procExtChoice*⟨⟨*Process* × *Process*⟩⟩ |
procIntChoice⟨⟨*Process* × *Process*⟩⟩ | *procPar*⟨⟨*Process* × *CSEExpression* × *Process*⟩⟩ |
procInter⟨⟨*Process* × *Process*⟩⟩ | *procHide*⟨⟨*Process* × *CSEExpression*⟩⟩ |
procRename⟨⟨*Process* × seq *N* × seq *N*⟩⟩ | *procParam*⟨⟨*Declaration* × *Process*⟩⟩ |
procInstP⟨⟨*Process* × seq *Expression*⟩⟩ | *procGeneric*⟨⟨seq *N* × *Process*⟩⟩ |
procInstG⟨⟨*Process* × seq *Expression*⟩⟩ |
procItrInter⟨⟨*Declaration* × *Process*⟩⟩

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

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

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

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

$ChannelDefinition == CDeclaration$

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

$CircusProgram == seq CircusParagraph$

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*⟩⟩ |
 apehType⟨⟨*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*

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

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

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

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

$ProgSafelet : SCJProgram \rightarrow Safelet$	$\forall s : SCJProgram$ $\bullet ProgSafelet(s) = s.1$
------------------------------------------------	------------------------------------------------------------

$ProgTLMS : SCJProgram \rightarrow TopLevelMissionSequencer$	$\forall s : SCJProgram$ $\bullet ProgTLMS(s) = s.2$
--------------------------------------------------------------	---------------------------------------------------------

$ProgTiers : SCJProgram \rightarrow \text{seq } Tier$	$\forall s : SCJProgram$ $\bullet ProgTiers(s) = s.3$
-------------------------------------------------------	----------------------------------------------------------

Framework

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

[*ID*]

[*Type*]

SafeletFWName : *N*
TopLevelMissionSequencerFWNMame : *N*

controlTierSync : *CSExpression*
Tier0 : *N*
MissionIds : seq *CircusParagraph*
SchedulableIds : seq *CircusParagraph*
ThreadIds : seq *CircusParagraph*
ObjectIds : seq *CircusParagraph*

ServicesChan : seq *CircusParagraph*
GlobalTypes : seq *CircusParagraph*
JTime : seq *CircusParagraph*
PrimitiveTypes : seq *CircusParagraph*
Priority : seq *CircusParagraph*
PriorityQueue : seq *CircusParagraph*
FrameworkChan : seq *CircusParagraph*
MissionId : seq *CircusParagraph*
SchedulableId : seq *CircusParagraph*

ObjectFW : *CircusParagraph*
ObjectChan : seq *CircusParagraph*
ObjectFWChan : seq *CircusParagraph*
ObjectMethChan : seq *CircusParagraph*
ThreadFW : *CircusParagraph*
ThreadChan : seq *CircusParagraph*
ThreadFWChan : seq *CircusParagraph*
ThreadMethChan : seq *CircusParagraph*

SafeletFW : *CircusParagraph*
SafeletFWChan : seq *CircusParagraph*
SafeletChan : seq *CircusParagraph*
SafeletMethChan : seq *CircusParagraph*

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

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

Build Phase

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

$TranslatablePrograms : \mathbb{P} SCJProgram$ $TranslatablePrograms \subset SCJProgram$ $\forall s : SCJProgram$ <ul style="list-style-type: none"> $s \in TranslatablePrograms$ $\Leftrightarrow ProgTLMS(s) \neq NoSequencer$ $\wedge ProgTiers(s) \neq \langle \rangle$ $\wedge \forall c : Cluster$ <ul style="list-style-type: none"> $\exists t : Tier; tiers : seq Tier$ <ul style="list-style-type: none"> $tiers = ProgTiers(s)$ $\wedge t \in ran tiers$ $\wedge c \in ran t$ $c.2 \neq \emptyset$
$AppEnv$ $Name : N$ $Parameters : seq Expression$
$ClusterAppEnv$ $Mission : AppEnv$ $Schedulables : \mathbb{F} AppEnv$ $Schedulables \neq \emptyset$
$TierAppEnv$ $Clusters : seq ClusterAppEnv$ $Clusters \neq \langle \rangle$
$AppProcEnv$ $Safelet : AppEnv$ $TopLevelMS : AppEnv$ $Tiers : seq TierAppEnv$ $Tiers \neq \langle \rangle$
$GetSafeletAppEnv : AppProcEnv \rightarrow AppEnv$ $\forall a : AppProcEnv$ <ul style="list-style-type: none"> $GetSafeletAppEnv(a) = a.Safelet$

$GetTLMSAppEnv : AppProcEnv \rightarrow AppEnv$	
$\forall a : AppProcEnv \bullet$ $GetTLMSAppEnv(a) = a.TopLevelMS$	
$GetTiersAppEnv : AppProcEnv \rightarrow seq\ TierAppEnv$	
$\forall a : AppProcEnv \bullet$ $GetTiersAppEnv(a) = a.Tiers$	
$IDof : Identifier \rightarrow N$	
$ParamsOf : seq\ ClassBodyDeclaration \rightarrow seq\ Expression$	
$BuildSOAppEnv : \mathbb{F}\ SchedulableObject \rightarrow \mathbb{F}\ AppEnv$	
$\forall scheds : dom\ BuildSOAppEnv$ $ scheds \neq \emptyset$ $\bullet \exists manT : ManagedThread; nestMS : NestedMissionSequencer; eh : EventHandler$ $perEH : PeriodicEventHandler; oneEH : OneShotEventHandler;$ $aephShort : Identifier \times EventHandlerClassBody;$ $aephLong : Identifier \times LongEventHandlerClassBody$ $\bullet BuildSOAppEnv(scheds) = \{a : AppEnv$ $ \forall so : scheds \bullet \exists name : N; params : seq\ Expression$ $ so = mt(manT) \Rightarrow$ $name = IDof(manT.1) \wedge params = ParamsOf(manT.2.2)$ $\wedge so = nms(nestMS) \Rightarrow$ $name = IDof(nestMS.1) \wedge params = ParamsOf(nestMS.2.2)$ $\wedge so = handler(pehDecl(perEH)) \Rightarrow$ $name = IDof(perEH.1) \wedge params = ParamsOf(perEH.2.2)$ $\wedge so = handler(osehDecl(oneEH)) \Rightarrow$ $name = IDof(oneEH.1) \wedge params = ParamsOf(oneEH.2.2)$ $\wedge so = handler(aephDecl(aephType(aephShort))) \Rightarrow$ $name = IDof(aephShort.1) \wedge params = ParamsOf(aephShort.2.2)$ $\wedge so = handler(aephDecl(aephType(aephLong))) \Rightarrow$ $name = IDof(aephLong.1) \wedge params = ParamsOf(aephLong.2.2)$ $\bullet a = \langle Name == name, Parameters == params \rangle\}$	
$BuildClusterAppEnv : Cluster \rightarrow ClusterAppEnv$	
$\forall c : dom\ BuildClusterAppEnv$ $ c.2 \neq \emptyset$ $\bullet \exists m : Mission; seqSO : \mathbb{F}\ SchedulableObject$ $ c = (m, seqSO)$ $\bullet BuildClusterAppEnv(c) =$ $\langle Mission == \langle Name == IDof(m.1), Parameters == ParamsOf(m.2.3) \rangle,$ $Schedulables == BuildSOAppEnv(seqSO) \rangle$	
$BuildClusterAppEnvs : seq\ Cluster \rightarrow seq\ ClusterAppEnv$	

$BuildTierAppEnv : Tier \rightarrow TierAppEnv$	
$\forall tier : dom\ BuildTierAppEnv$ $ tier \neq \langle \rangle$ $\bullet BuildTierAppEnv(tier) = \langle Clusters == BuildClusterAppEnvs(tier) \rangle$	
$BuildTiersAppEnv : seq\ Tier \rightarrow seq\ TierAppEnv$	
$\forall tiers : dom\ BuildTiersAppEnv$ $ tiers \neq \langle \rangle$ $\bullet \# tiers = 1 \Rightarrow BuildTiersAppEnv(tiers) = \langle BuildTierAppEnv(head\ tiers) \rangle$ $\wedge \# tiers \geq 1 \Rightarrow BuildTiersAppEnv(tiers) =$ $\quad \langle BuildTierAppEnv(head\ tiers) \rangle \frown BuildTiersAppEnv(tail\ tiers)$	
$BuildAppProcEnv : SCJProgram \rightarrow AppProcEnv$	
$\forall scjProg : dom\ BuildAppProcEnv$ $ scjProg \in TranslatablePrograms$ $\bullet \exists safelet : Safelet; tiers : seq\ Tier; ms : MissionSequencer$ $ scjProg = (safelet, tlms(ms), tiers)$ $\bullet \exists sfEnv : AppEnv; tlmsEnv : AppEnv;$ $\quad tiersEnv : seq\ TierAppEnv$ $\bullet sfEnv = \langle Name == IDof(safelet.1),$ $\quad \quad Parameters == ParamsOf(safelet.2.4) \rangle$ $\wedge tlmsEnv = \langle Name == IDof(ms.1),$ $\quad \quad Parameters == ParamsOf(ms.2.2) \rangle$ $\wedge tiersEnv = BuildTiersAppEnv(tiers)$ $\wedge BuildAppProcEnv(scjProg) = \langle Safelet == sfEnv,$ $\quad \quad TopLevelMS == tlmsEnv, Tiers == tiersEnv \rangle$	
$LockingEnv$	
$Threads : seq(ThreadIds \times Priority)$ $Objects : seq\ ObjectIds$	
$Threads \neq \langle \rangle$ $Objects \neq \langle \rangle$	
$BuildLockEnv : SCJProgram \rightarrow LockingEnv$	
$\forall scjProg : dom\ BuildLockEnv$ $ scjProg \in TranslatablePrograms$ $\bullet \exists lockEnv : LockingEnv$ $\bullet BuildLockEnv(scjProg) = lockEnv$	

ClusterEnv

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

$\text{disjoint}(\text{NestedMissionSequencers}, \text{ManagedThreads}, \text{PeriodicEventHandlers},$
 $\text{AperiodicEventHandlers}, \text{OneShotEventHandlers})$
 $\bigcup\{\text{NestedMissionSequencers}, \text{ManagedThreads}, \text{PeriodicEventHandlers},$
 $\text{AperiodicEventHandlers}, \text{OneShotEventHandlers}\} \neq \emptyset$

TierEnv

Clusters : seq *ClusterEnv*

Clusters $\neq \langle \rangle$

FWEnv

TopLevelMS : Identifier

Tiers : seq *TierEnv*

Tiers $\neq \langle \rangle$

GetTierFWEnvs : *FWEnv* \rightarrow seq *TierEnv*

$\forall \text{env} : \text{FWEnv}$

• $\text{GetTierFWEnvs}(\text{env}) = \text{env.Tiers}$

GetIdentifiers : \mathbb{F} *SchedulableObject* \leftrightarrow \mathbb{F} Identifier

$\forall \text{scheds} : \text{dom } \text{GetIdentifiers}$

| $\text{scheds} \neq \emptyset$

• $\exists \text{manT} : \text{ManagedThread}; \text{nestMS} : \text{NestedMissionSequencer};$
 $\text{perEH} : \text{PeriodicEventHandler}; \text{oneEH} : \text{OneShotEventHandler};$
 $\text{eh} : \text{EventHandler};$

$\text{apehShort} : \text{Identifier} \times \text{EventHandlerClassBody};$

$\text{apehLong} : \text{Identifier} \times \text{LongEventHandlerClassBody}$

• $\text{GetIdentifiers}(\text{scheds}) = \{i : \text{Identifier}$

| $\forall s : \text{scheds}$

| $\text{scheds} \neq \emptyset$

• $s = \text{mt}(\text{manT}) \Rightarrow i = \text{manT}.1$

$\wedge s = \text{nms}(\text{nestMS}) \Rightarrow i = \text{nestMS}.1$

$\wedge s = \text{handler}(\text{pehDecl}(\text{perEH})) \Rightarrow i = \text{perEH}.1$

$\wedge s = \text{handler}(\text{apehDecl}(\text{apehType}(\text{apehShort}))) \Rightarrow i = \text{apehShort}.1$

$\wedge s = \text{handler}(\text{apehDecl}(\text{apehType}(\text{apehLong}))) \Rightarrow i = \text{apehLong}.1$

$\wedge s = \text{handler}(\text{osehDecl}(\text{oneEH})) \Rightarrow i = \text{oneEH}.1$

}

$BuildSOEnvs : \mathbb{F} \text{SchedulableObject} \rightarrow$

$\mathbb{F} \text{Identifier} \times \mathbb{F} \text{Identifier} \times \mathbb{F} \text{Identifier} \times$
 $\mathbb{F} \text{Identifier} \times \mathbb{F} \text{Identifier}$

$\forall s : \text{dom } BuildSOEnvs$

$| s \neq \emptyset$

- $\exists sms : \mathbb{F} \text{Identifier}; pehs : \mathbb{F} \text{Identifier};$
 $aehs : \mathbb{F} \text{Identifier}; oehs : \mathbb{F} \text{Identifier}; mts : \mathbb{F} \text{Identifier}$
 $| mts = GetIdentifiers(\{mtSched : s$
 $| \exists m : \text{ManagedThread}$
 $\bullet mtSched = mt(m)\})$
 $\wedge sms = GetIdentifiers(\{nmsSched : s$
 $| \exists n : \text{NestedMissionSequencer}$
 $\bullet nmsSched = nms(n)\})$
 $\wedge pehs = GetIdentifiers(\{pehSched : s$
 $| \exists p : \text{PeriodicEventHandler}$
 $\bullet pehSched = handler(pehDecl(p))\})$
 $\wedge aehs = GetIdentifiers(\{aehSched : s$
 $| \exists a : \text{Identifier} \times \text{EventHandlerClassBody}$
 $\bullet aehSched = handler(aehDecl(aehType(a)))\})$
 $\wedge aehs = GetIdentifiers(\{aehLSched : s$
 $| \exists a : \text{Identifier} \times \text{LongEventHandlerClassBody}$
 $\bullet aehLSched = handler(aehDecl(aehType(a)))\})$
 $\wedge oehs = GetIdentifiers(\{osehSched : s$
 $| \exists o : \text{OneShotEventHandler}$
 $\bullet osehSched = handler(osehDecl(o))\})$
 $\bullet BuildSOEnvs(s) = (sms, pehs, aehs, oehs, mts)$

$BuildClusterEnv : \text{Cluster} \rightarrow \text{ClusterEnv}$

$\forall c : \text{dom } BuildClusterEnv$

$| c.2 \neq \emptyset$

- $\exists missionName : \text{Identifier}; sms : \mathbb{F} \text{Identifier}; pehs : \mathbb{F} \text{Identifier};$
 $aehs : \mathbb{F} \text{Identifier}; oseh : \mathbb{F} \text{Identifier}; mts : \mathbb{F} \text{Identifier}; cluster : \text{ClusterEnv}$
 $| missionName = c.1.1$
 $\wedge (sms, pehs, aehs, oseh, mts) = BuildSOEnvs(c.2)$
 $\bullet BuildClusterEnv(c) =$
 $\langle Mission == missionName, NestedMissionSequencers == sms, PeriodicEventHandlers == pehs,$
 $AperiodicEventHandlers == aehs, OneShotEventHandlers == oseh, ManagedThreads == mts \rangle$

$BuildClusterEnvs : \text{seq Cluster} \rightarrow \text{seq ClusterEnv}$

$\forall c : \text{dom } BuildClusterEnvs$

$| c \neq \langle \rangle \wedge \forall s : \text{seq Cluster} \bullet s \neq \langle \rangle$

- $\# c = 1 \Rightarrow BuildClusterEnvs(c) = \langle BuildClusterEnv(head\ c) \rangle$
 $\wedge \# c \geq 1 \Rightarrow BuildClusterEnvs(c) = \langle BuildClusterEnv(head\ c) \rangle \frown BuildClusterEnvs(tail\ c)$

$BuildTierEnv : \text{Tier} \rightarrow \text{TierEnv}$

$\forall tier : \text{seq Cluster}$

- $BuildTierEnv(tier) = \langle Clusters == BuildClusterEnvs(tier) \rangle$

$BuildTierEnvs : seq\ Tier \rightarrow seq\ TierEnv$	
$\forall tiers : seq\ Tier \bullet$ $BuildTierEnvs(tiers) = \langle BuildTierEnv(head\ tiers) \rangle \frown BuildTierEnvs(tail\ tiers)$	
$BuildFWEnv : SCJProgram \rightarrow FWEnv$	
$\forall scjProg : dom\ BuildFWEnv$ $\quad scjProg \in TranslatablePrograms$ $\quad \bullet \exists tlms : MissionSequencer; tlmsID : Identifier; tlmsBody : MissionSequencerClassBody;$ $\quad tiers : seq\ Tier \mid$ $\quad \quad scjProg.2 \neq NoSequencer \Rightarrow tlms = (tlmsID, tlmsBody)$ $\quad \quad \wedge tiers = scjProg.3 \bullet$ $\quad \quad BuildFWEnv(scjProg) =$ $\quad \quad \langle TopLevelMS == tlms.1, Tiers == BuildTierEnvs(tiers) \rangle$	
$BinderMethodEnv$	
$MethodName : N$	
$Locs : \mathbb{F}\ N$	
$Callers : \mathbb{F}\ N$	
$ReturnType : Type$	
$Params : seq\ Type$	
$Synchronised : \mathbb{B}$	
$LocParam : N$	
$LocType : Type$	
$CallerType : Type$	
$MCBEnv$	
$BinderMethods : seq\ BinderMethodEnv$	
$BinderMethods \neq \langle \rangle$	
$GetSFMethods : Safelet \rightarrow seq\ ClassBodyDeclaration$	
$\forall sf : Safelet$ $\quad \bullet GetSFMethods(sf) = sf.2.4$	
$GetTLMSMethods : MissionSequencer \rightarrow seq\ ClassBodyDeclaration$	
$\forall tlms : MissionSequencer$ $\quad \bullet GetTLMSMethods(tlms) = tlms.2.2$	
$BuildBME : N \times N \times MethodDeclaration \rightarrow BinderMethodEnv$	
$BuildSFMCBEnv : seq\ ClassBodyDeclaration \rightarrow seq\ BinderMethodEnv$	
$\forall body : dom\ BuildSFMCBEnv$ $\quad body \neq \langle \rangle \wedge \forall b : seq\ ClassBodyDeclaration \bullet b \neq \langle \rangle$ $\quad \bullet BuildSFMCBEnv(body) = \langle \rangle$	

$BuildTLMSMCBEnv : seq\ ClassBodyDeclaration \rightarrow seq\ BinderMethodEnv$

$BuildTierMCBEnv : seq\ Tier \rightarrow seq\ BinderMethodEnv$

$BuildMCBEnv : SCJProgram \mapsto seq\ BinderMethodEnv$

$\forall\ scjProg : dom\ BuildMCBEnv$
 $\quad | \ scjProg \in TranslatablePrograms$
 $\quad \bullet \exists\ sf : Safelet; \ tlms : MissionSequencer; \ tiers : seq\ Tier$
 $\quad \quad | \ sf = scjProg.1$
 $\quad \quad \wedge \ tiers = scjProg.3$
 $\quad \bullet \ BuildMCBEnv(scjProg) =$
 $\quad \quad BuildSFMCBEnv(GetSFMethods(sf))$
 $\quad \quad \frown BuildTLMSMCBEnv(GetTLMSMethods(tlms))$
 $\quad \quad \frown BuildTierMCBEnv(tiers)$

Generate Phase

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

$procNameOf : Process \rightarrow N$

$ControlTierSync : CSEExpression$

$MissionSync : CSEExpression$

$SchedulablesSync : CSEExpression$

$TierSync : TierEnv \rightarrow CSEExpression$

$\forall t : TierEnv$

- $\exists m : seq\ N$
- $TierSync(t) = cs(m)$

$GetMissionID : ClusterEnv \rightarrow N$

$GenerateTiersFWProc : ClusterEnv \rightarrow Process$

$GenerateClusterFWProcs : seq\ ClusterEnv \rightarrow Process$

$\forall clusters : dom\ GenerateClusterFWProcs$

| $clusters \neq \langle \rangle$

- $\# clusters = 1$

$\Rightarrow GenerateClusterFWProcs(clusters) =$

$procPar(\$
 $procName(GetMissionID(head\ clusters)),$
 $MissionSync,$
 $GenerateTiersFWProc(head\ clusters)$
 $\)$

$\wedge \# clusters \geq 1$

$\Rightarrow GenerateClusterFWProcs(clusters) =$

$procPar(\$
 $procPar(\$
 $procName(GetMissionID(head\ clusters)),$
 $MissionSync,$
 $GenerateTiersFWProc(head\ clusters)),$
 $SchedulablesSync,$
 $GenerateClusterFWProcs(tail\ clusters)$
 $\)$

$GenerateTierFWProcs : seq\ TierEnv \rightarrow seq\ Process$

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

$GenerateTierFWProc : seq\ TierEnv \rightarrow Process$

$ControlTier : N$
 $TopLevelMissionSequencerFWName : N$

$GetParams : Identifier \rightarrow seq\ Expression$

$GenerateFWProcs : FWEnv \rightarrow seq\ Process$

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

$GenerateAppTierProcs : seq\ TierAppEnv \rightarrow Process$

$GenerateAppProc : AppProcEnv \rightarrow Process$

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

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

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

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

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

$BinderRetChan : N \rightarrow \text{seq } N$

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

$GenerateMCBChan : BinderMethodEnv \rightarrow CircusParagraph$

$\forall bme : BinderMethodEnv$
 $\bullet \text{ GenerateMCBChan}(bme) = \text{chanDef}(\text{multiDecl}(\text{chanNameWithType}(\text{NaturalCallChan}(bme.MethodName), \text{MCBParams}(bme.Params)), \text{scDecl}(\text{chanNameWithType}(\text{NaturalRetChan}(bme.MethodName), \text{MCBParams}(bme.Params))))))$

$MethodCallBinderSync : N$

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

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

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

$BinderCallComm : N \rightarrow N$

$NaturalCallComm : N \rightarrow N$

$NaturalRetComm : N \rightarrow N$

$BindeRetComm : N \rightarrow N$

$GenerateMCBName : N \rightarrow N$

$BinderCallParams : seq\ Type \rightarrow seq\ CParameter$

$NaturalCallParams : seq\ Type \rightarrow seq\ CParameter$

$NaturalRetParams : seq\ Type \rightarrow seq\ CParameter$

$BinderRetParams : seq\ Type \rightarrow seq\ CParameter$

$BinderActions : N$

$DoneTLS : Communication$

$NoState : SchemaExp$

$MethodCallBinder : N$

$GenerateMCBAction : BinderMethodEnv \rightarrow PParagraph$

$\forall bme : BinderMethodEnv$

- $GenerateMCBAction(bme) = actDef(GenerateMCBName(bme.MethodName),$
 $prefixExp((BinderCallComm(bme.MethodName),$
 $BinderCallParams(bme.Params)),$
 $prefixExp((NaturalCallComm(bme.MethodName),$
 $BinderCallParams(bme.Params)),$
 $prefixExp((NaturalRetComm(bme.MethodName),$
 $BinderCallParams(bme.Params)),$
 $prefixExp((BindeRetComm(bme.MethodName),$
 $BinderCallParams(bme.Params)),$
 $actName(GenerateMCBName(bme.MethodName))$
 $)$
 $)$
 $)$
 $)$
 $)$
 $)$

$GenerateMCBActions : seq\ BinderMethodEnv \rightarrow seq\ PParagraph$

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

$GenerateMCBProc : seq\ BinderMethodEnv \rightarrow CircusParagraph$

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

$GenerateMCBModel : seq\ BinderMethodEnv \rightarrow seq\ CircusParagraph$

$\forall bEnvs : seq\ BinderMethodEnv$
 $| bEnvs \neq \langle \rangle$
 $\bullet GenerateMCBModel(bEnvs) = GenerateMCBChans(bEnvs) \wedge$
 $\quad \langle GenerateMethodCallBinderSync(bEnvs), GenerateMCBProc(bEnvs) \rangle$

$GenerateThreadProc : seq(ThreadIds \times Priority) \rightarrow Process$

$GenerateObjectProc : seq\ ObjectIds \rightarrow Process$

$GenerateLockModel : LockingEnv \rightarrow seq\ CircusParagraph$

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

Translate SCJ Program

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

ProcessID : $N \rightarrow ID$

TransClasses : *SCJProgram* \rightarrow *CircusProgram*

FWName : N

AppName : N

MCBName : N

LockName : N

ProgName : *Identifier* $\rightarrow N$

appComms : *CSExpression*

mcbComms : *CSExpression*

lockComms : *CSExpression*

TransSCJProg : *Identifier* \times *SCJProgram* \rightarrow *CircusProgram*

\forall *scjProg* : *SCJProgram*; *name* : *Identifier* •
 \exists *app* : *CircusProgram*;
program : *CircusProgram*;
fwProcs : seq *Process*; *appProc* : *Process*; *lockModel* : seq *CircusParagraph*;
mcbModel : seq *CircusParagraph* |
app = *TransClasses*(*scjProg*) \wedge
fwProcs = *GenerateFWProcs*(*BuildFWEnv*(*scjProg*)) \wedge
appProc = *GenerateAppProc*(*BuildAppProcEnv*(*scjProg*)) \wedge
mcbModel = *GenerateMCBModel*(*BuildMCBEnv*(*scjProg*)) \wedge
lockModel = *GenerateLockModel*(*BuildLockEnv*(*scjProg*)) \wedge
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))))) •
TransSCJProg(*name*, *scjProg*) =
framework \wedge \langle procDef(pd(*FWName*, head *fwProcs*)) \rangle \wedge
app \wedge \langle procDef(pd(*AppName*, *appProc*)) \rangle \wedge
mcbModel \wedge
lockModel \wedge
program

Low Level

- $Method : MethodDeclaration \mapsto (Name, Params, ReturnType, Body) :$ translates an active application method into a *Circus* action
- $DataMethod : MethodDeclaration \mapsto :$ translates data methods into an *OhCircus* method
- $MethodBody : Block \mapsto \text{seq } CircExpression :$ translates a Java block, for example a method body
- $Registers : Block \mapsto \text{seq } Name :$ extracts the Names of the schedulables registered in a Java block
- $Returns : Block \mapsto \text{seq } Name :$ extracts the Names of the variables returned in a Java block
- $Variable : (Name, Type, InitExpression) \mapsto (CircName, CircType, CircExpression) :$ translates a variable
- $Parameters : (Name, Params, ReturnType, Body) \mapsto \text{seq } CircParam :$ translates a list of method parameters
- $\llbracket Name \rrbracket_{name} :$ translates the *name* to a Z identifier
- $\llbracket varType \rrbracket_{type} :$ translates types
- $\llbracket expr \rrbracket_{expression} :$ 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 $\llbracket Identifier \rrbracket_{Name} App \hat{=} \llbracket \llbracket ConstructorDeclaration \rrbracket_{Method} \rrbracket_{Parameters} \mathbf{begin}$

State
 $this : \mathbf{ref} \llbracket Identifier \rrbracket_{name} Class$

state *State*

Init
 $State'$
 $this := \mathbf{new} \llbracket Identifier \rrbracket_{name} Class()$

$InitializeApplication \hat{=}$

$$\left(\begin{array}{l} initializeApplicationCall \longrightarrow \\ \llbracket \llbracket InitializeApplication \rrbracket_{Method} \rrbracket_{MethBody} \\ initializeApplicationRet \longrightarrow \\ \mathbf{Skip} \end{array} \right)$$

$GetSequencer \hat{=}$

$$\left(\begin{array}{l} getSequencerCall \longrightarrow \\ getSequencerRet ! \llbracket GetSequencer \rrbracket_{Returns} \longrightarrow \\ \mathbf{Skip} \end{array} \right)$$

$\llbracket AppMeth_1 \rrbracket_{Method}$

...

$\llbracket AppMeth_n \rrbracket_{Method}$

$$Methods \hat{=} \left(\begin{array}{l} GetSequencer \\ \square \\ InitializeApplication \\ \square \\ MethName(AppMeth_1) \\ \square \\ \dots \\ MethName(AppMeth_n) \\ \dots \end{array} \right) ; Methods$$

- $(Init ; Methods) \triangle (end_safelet_app \longrightarrow \mathbf{Skip})$

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 }

```

process $\llbracket Identifier \rrbracket_{Name} App \hat{=} \llbracket \llbracket ConstructorDeclaration \rrbracket_{Method} \rrbracket_{Parameters}$ **begin**

State

this : ref $\llbracket Identifier \rrbracket_{name} Class$

state *State*

Init

State'
this := **new** $\llbracket Identifier \rrbracket_{name} Class()$

GetNextMission $\hat{=} \mathbf{var} \text{ ret} : MissionID \bullet$

$$\left(\begin{array}{l} getNextMissionCall . IdOf(Identifier) \longrightarrow \\ ret := this . getNextMission(); \\ getNextMissionRet . IdOf(Identifier) ! ret \longrightarrow \\ \mathbf{Skip} \end{array} \right)$$

$\llbracket AppMeth_1 \rrbracket_{Method}$

...

$\llbracket AppMeth_n \rrbracket_{Method}$

Methods $\hat{=}$

$$\left(\begin{array}{l} GetNextMission \\ \square \\ MethName(AppMeth_1) \\ \square \\ MethName(AppMeth_n) \\ \dots \end{array} \right); Methods$$

$\bullet (Init ; Methods) \triangle (end_sequencer_app . IdOf(Identifier) \longrightarrow \mathbf{Skip})$

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 $\llbracket Identifier \rrbracket App \hat{=} \llbracket \llbracket ConstructorDeclaration \rrbracket_{Method} \rrbracket_{Parameters}$ **begin**

State
 $this : \text{ref } \llbracket Identifier \rrbracket_{name} Class$

state *State*

Init
 $State'$
 $this := \text{new } \llbracket Identifier \rrbracket_{name} Class()$

InitializePhase $\hat{=}$

$$\left(\begin{array}{l} initializeCall . IdOf(Identifier) \longrightarrow \\ \llbracket initialize \rrbracket_{Registers} initializeRet . IdOf(Identifier) \longrightarrow \\ \text{Skip} \end{array} \right)$$

CleanupPhase $\hat{=}$

$$\left(\begin{array}{l} cleanupMissionCall . IdOf(Identifier) \longrightarrow \\ cleanupMissionRet . IdOf(Identifier) ! \text{True} \longrightarrow \\ \text{Skip} \end{array} \right)$$

$\llbracket AppMeth_1 \rrbracket_{Method}$

...

$\llbracket AppMeth_n \rrbracket_{Method}$

$Methods \hat{=}$
$$\left(\begin{array}{l} InitializePhase \\ \square \\ CleanupPhase \\ \square \\ MethName(AppMeth_1) \\ \square \\ MethName(AppMeth_n) \\ \dots \end{array} \right) ; Methods$$

• $(Init ; Methods) \triangle (end_mission_app . IdOf(Identifier) \longrightarrow \mathbf{Skip}$

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 }

```

process $\llbracket PName \rrbracket App \hat{=} \llbracket \llbracket ConstructorDeclaration \rrbracket_{Method} \rrbracket_{Parameters}$ **begin**

State
this : ref $\llbracket Identifier \rrbracket_{name} Class$

state *State*

Init
State '
this := **new** $\llbracket Identifier \rrbracket_{name} Class()$

handleAsyncEvent $\hat{=}$

$$\left(\begin{array}{l} handleAsyncEventCall . IdOf(PName) \longrightarrow \\ \llbracket \llbracket HandleAsyncBody \rrbracket_{Method} \rrbracket_{MethBody}; \\ handleAsyncEventRet . IdOf(PName) \longrightarrow \\ \mathbf{Skip} \end{array} \right)$$

$\llbracket AppMeth_1 \rrbracket_{Method}$

...

$\llbracket AppMeth_n \rrbracket_{Method}$

Methods $\hat{=}$

$$\left(\begin{array}{l} handleAsyncEvent \\ \square \\ MethName(AppMeth_1) \\ \square \\ MethName(AppMeth_n) \\ \dots \end{array} \right); Methods$$

• (*Init* ; *Methods*) $\triangle (end_ \llbracket HandlerTypeIdOf(PName) \rrbracket \longrightarrow \mathbf{Skip})$

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 }

```

process $\llbracket PName \rrbracket App \hat{=} \llbracket \llbracket ConstructorDeclaration \rrbracket_{Method} \rrbracket_{Parameters}$ **begin**

State
 $this : \text{ref } \llbracket Identifier \rrbracket_{name} Class$

state *State*

Init
 $State'$
 $this := \text{new } \llbracket Identifier \rrbracket_{name} Class()$

$Run \hat{=}$

$$\left(\begin{array}{l} runCall . IdOf(PName) \longrightarrow \\ \llbracket \llbracket run \rrbracket_{Method} \rrbracket_{MethBody}; \\ runRet . IfOf(PName) \longrightarrow \\ \text{Skip} \end{array} \right)$$

$\llbracket AppMeth_1 \rrbracket_{Method}$

...

$\llbracket AppMeth_n \rrbracket_{Method}$

$Methods \hat{=}$

$$\left(\begin{array}{l} Run \\ \square \\ MethName(AppMeth_1) \\ \square \\ MethName(AppMeth_n) \\ \dots \end{array} \right); Methods$$

• $(Init ; Methods) \triangle (end_managedThread_app . IdOf(PName) \longrightarrow \text{Skip})$

end

Data Class

class $\llbracket PName \rrbracket_{name}$ *Class* $\hat{=}$ **begin**

state *State*

$\llbracket VarName \rrbracket_{name} : \llbracket VarType \rrbracket_{type}$

state *State*

initial *Init*

State '

$\llbracket VarName \rrbracket'_{name} = \llbracket VarInit \rrbracket_{expression}$

$\llbracket DataMeth1 \rrbracket_{dataMeth}$

$\llbracket DataMeth2 \rrbracket_{dataMeth}$

...

• **Skip**

end