Translation Rules

High Level

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$$ChannelDefinition == CDeclaration$$

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

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

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

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

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

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

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

$NestedMissionSequencer == MissionSequencer$

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

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

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

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

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

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

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

[*ID*]

[*Type*]

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

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

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

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

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

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

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

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

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

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

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

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

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

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

$framework : CircusProgram$

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

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

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

$$TierAppEnv == \text{seq}(\text{seq}(AppEnv \times \text{seq}\, AppEnv))$$

$$AppProcEnv == AppEnv \times AppEnv \times TierAppEnv$$

$$GetSafeletAppEnv : AppProcEnv \pfun AppEnv$$

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

$$GetTLMSAppEnv : AppProcEnv \pfun AppEnv$$

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

$$GetTierAppEnvs : AppProcEnv \pfun TierAppEnv$$

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

$$IDof : Identifier \pfun N$$

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

$$BuildSOAppEnv : \text{seq}\, SchedulableObject \pfun \text{seq}\, AppEnv$$

$$BuildClusterAppEnv : Cluster \pfun AppEnv \times \text{seq}\, AppEnv$$

$$\forall c : Cluster \bullet \exists m : Mission;\ seqSO : \text{seq}\, SchedulableObject \mid c = (m, seqSO) \bullet BuildClusterAppEnv(c) = ((IDof(m.1), ParamsOf(m.2.3)), BuildSOAppEnv(seqSO))$$

$$BuildTierAppEnv : Tier \pfun \text{seq}(AppEnv \times \text{seq}\, AppEnv)$$

$$\forall t : Tier \bullet \exists c : Cluster;\ seqC : \text{seq}\, Cluster \mid t = \langle c \rangle \frown seqC \bullet BuildTierAppEnv(t) = \langle BuildClusterAppEnv(c) \rangle \frown BuildTierAppEnv(seqC)$$

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

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

$$BuildAppProcEnv : SCJProgram \pfun AppProcEnv$$

$$\forall\, scjProg : SCJProgram \bullet \exists\, safelet : Safelet;\ topLevelMs : TopLevelMissionSequencer;\ tiers : \text{seq}\ Tier \mid scjProg = (safelet, topLevelMs, tiers) \bullet \exists\, sfEnv : AppEnv;\ tlmsEnv : AppEnv;\ tiersEnv : TierAppEnv;\ ms : MissionSequencer \bullet sfEnv = (IDof(safelet.1), ParamsOf(safelet.2.4)) \land topLevelMs = tlms(ms) \Rightarrow tlmsEnv = (IDof(ms.1), ParamsOf(ms.2.2)) \land tiersEnv = BuildTiersAppEnv(tiers) \land BuildAppProcEnv(scjProg) = (sfEnv, tlmsEnv, tiersEnv)$$

$$LockingEnv == \text{seq}\ ThreadIds \times \text{seq}\ ObjectIds$$

$$BuildLockEnv : SCJProgram \pfun LockingEnv$$

$$SplitSchedulableObjects : \text{seq}\ SchedulableObject \pfun \text{seq}\ SchedulableObject \times \text{seq}\ SchedulableObject \times \text{seq}\ SchedulableObject \times \text{seq}\ SchedulableObject \times \text{seq}\ SchedulableObject$$

$$BuildSOEnvs : \text{seq}\ SchedulableObject \pfun \text{seq}\ Identifier \times \text{seq}\ Identifier \times \text{seq}\ Identifier \times \text{seq}\ Identifier \times \text{seq}\ Identifier$$

$$\forall\, s : \text{seq}\ SchedulableObject \bullet \exists\, so : SchedulableObject;\ sms : \text{seq}\ Identifier;\ pehs : \text{seq}\ Identifier;\ apehs : \text{seq}\ Identifier;\ oseh : \text{seq}\ Identifier;\ mts : \text{seq}\ Identifier \mid \exists\, n : NestedMissionSequencer \bullet head\ s = nms(n) \Rightarrow sms = sms \frown \langle n.1\rangle \bullet BuildSOEnvs(s) = (sms, pehs, apehs, oseh, mts)$$

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

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

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

$$BuildClusterEnv : Cluster \pfun ClusterEnv$$

$$\forall\, c : Cluster \bullet$$
$$\quad \exists\, missionName : Identifier;\ sms : \seq Identifier;\ pehs : \seq Identifier;$$
$$\qquad apehs : \seq Identifier;\ oseh : \seq Identifier;\ mts : \seq Identifier \mid$$
$$\quad missionName = c.1.1 \land$$
$$\quad (sms, pehs, apehs, oseh, mts) = BuildSOEnvs(c.2) \bullet$$
$$\quad BuildClusterEnv(c) = (missionName, sms, pehs, apehs, oseh, mts)$$

$$BuildTierEnv : Tier \pfun TierEnv$$

$$\forall\, t : \seq Cluster \bullet$$
$$\quad BuildTierEnv(t) = \langle BuildClusterEnv(head\ t)\rangle \frown BuildTierEnv(tail\ t)$$

$$BuildTierEnvs : \seq Tier \pfun \seq TierEnv$$

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

$$BuildFWEnv : SCJProgram \pfun FWEnv$$

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

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

$$MCBEnv == \seq BinderMethodEnv$$

$$BuildMCBEnv : SCJProgram \pfun \seq BinderMethodEnv$$

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

$$GenerateTierProcs : \text{seq}\ Tier \pfun Process$$

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

$$GenerateAppTierProcs : TierAppEnv \pfun Process$$

$$GenerateAppProc : AppProcEnv \pfun Process$$

$$\begin{aligned}
&\forall\, appProcEnv : AppProcEnv \bullet \\
&\quad \exists\, sfAppEnv : AppEnv;\ tlmsAppEnv : AppEnv;\ tiersAppEnvs : TierAppEnv \mid \\
&\quad sfAppEnv = GetSafeletAppEnv(appProcEnv) \land \\
&\quad tlmsAppEnv = GetTLMSAppEnv(appProcEnv) \bullet \\
&\quad GenerateAppProc(appProcEnv) = \\
&\qquad procInter(\\
&\qquad\quad procInter(\\
&\qquad\qquad procInstP(procName(sfAppEnv.1), sfAppEnv.2), \\
&\qquad\qquad procInstP(procName(tlmsAppEnv.1), tlmsAppEnv.2) \\
&\qquad\quad), \\
&\qquad\quad GenerateAppTierProcs(tiersAppEnvs) \\
&\qquad)
\end{aligned}$$

$$GenerateMCBProc : \text{seq}\ BinderMethodEnv \pfun Process$$

$$GenerateLockProc : LockingEnv \pfun \text{seq}\ CircusParagraph$$

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

$$ProcessID : N \pfun ID$$

$$TransClasses : SCJProgram \pfun CircusProgram$$

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

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

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

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 }
```

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

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 }
```

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

State

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

state *State*

Init

$State'$

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

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

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

...

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

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

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

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