

CHANGES NEEDED IN JCIRCUS:

– Implementation of pattern for axiomatic definition of \mathbb{P} type of k – Check if single sync in an external choice
Generic and dotted channels

$AreaId ::= A0 \mid A1$

$ZoneId ::= Z0 \mid Z1 \mid Z2 \mid Z3 \mid Z4 \mid Z5$

$Mode ::= automatic \mid manual \mid disabled$

$OnOff ::= on \mid off$

$AlarmStage ::= alarmOff \mid firstStage \mid secondStage$

$LampId ::= zoneFaultLamp$
 $\mid earthFaultLamp$
 $\mid sounderLineFaultLamp$
 $\mid powerFaultLamp$
 $\mid isolateRemoteSignalLamp$
 $\mid actuatorLineFaultLamp$
 $\mid circuitFaultLamp$
 $\mid alarmSilencedLamp$
 $\mid systemOnLamp$

$TId ::= zoneFaultLamp$
 $\mid earthFaultLamp$
 $\mid sounderLineFaultLamp$
 $\mid powerFaultLamp$
 $\mid isolateRemoteSignalLamp$
 $\mid actuatorLineFaultLamp$
 $\mid circuitFaultLamp$
 $\mid alarmSilencedLamp$
 $\mid systemOnLamp$
 $\mid A0 \mid A1 \mid Z0 \mid Z1 \mid Z2 \mid Z3 \mid Z4 \mid Z5$

$FaultId ::= zoneFault$
 $\mid earthFault$
 $\mid sounderLineFault$
 $\mid powerFault$
 $\mid isolateRemoteSignal$
 $\mid actuatorLineFault$

SystemState ::= *fireSysStarts*

| *fireSyss*
 | *fireSysDs*
 | *autos*
 | *countdowns*
 | *discharges*
 | *resets*
 | *manuals*
 | *disableds*

Bool ::= *ttrue* | *ffalse*

$getZones : AreaId \rightsquigarrow \mathbb{P} ZoneId$	$\frac{}{(getZones\ A0) = \{ Z0, Z1 \} \wedge (getZones\ A1) = \{ Z2, Z3 \}}$
--	---

$getLampId : FaultId \rightsquigarrow LampId$	$\frac{}{getLampId\ zoneFault = zoneFaultLamp \wedge$ $getLampId\ earthFault = earthFaultLamp \wedge$ $getLampId\ sounderLineFault = sounderLineFaultLamp \wedge$ $getLampId\ powerFault = powerFaultLamp \wedge$ $getLampId\ isolateRemoteSignal = isolateRemoteSignalLamp \wedge$ $getLampId\ actuatorLineFault = actuatorLineFaultLamp}$
---	--

$gasDelay : \mathbb{N}$	$\frac{}{gasDelay = 30}$
-------------------------	--------------------------

channel*switchOn, silenceAlarm, startClock, clockFinished*

channel*detection : ZoneId*

Obs: era SwitchMode

channel*modeSwitch : Mode*

channel*externalManualDischarge : $\mathbb{P} AreaId$*

channel*fault* : *FaultId*

channel*reset, actuatorsReplaced*

channel*alarm* : *AlarmStage*

channel*switchLamp* : *TId* \times *OnOff*

channel*switchBuzzer* : *OnOff*

channel*systemState* : *SystemState*

channel*displayDischarge, manualDischarge* : \mathbb{P} *AreaId*

channel*switched, automaticDischarge,*
anyDischarge, noDischarge, countdown, counting

channel*gasDischarged, gasNotDischarged* : *AreaId*

channelset *GasDischargeSync* ==
{
 manualDischarge, automaticDischarge, countdown, counting,
 gasDischarged, gasNotDischarged, anyDischarge, noDischarge
}

channelset *Sigma2* == {
 switchOn, reset, detection, modeSwitch, silenceAlarm,
 actuatorsReplaced,
 manualDischarge, automaticDischarge, countdown, counting,
 gasDischarged, gasNotDischarged, anyDischarge, noDischarge
}

channelset *Sigmaareas* == {
 switchOn, reset, modeSwitch, detection,
 silenceAlarm, actuatorsReplaced,
 automaticDischarge, manualDischarge,
 anyDischarge, noDischarge, counting }

channelset *DisplaySync* == { | *displayDischarge*, *switched* | }

channelset *Sigma1* == { |
 switchOn, *reset*, *detection*, *displayDischarge*,
 silenceAlarm, *actuatorsReplaced*, *fault* | }

process *FireControl* $\hat{=}$ **begin**

state *InternalSystemState* == [*mode1* : *Mode*]

InitFireControl $\hat{=}$ *mode1* := *automatic*

SwitchFireControlMode $\hat{=}$ *newMode* : *Mode* • *mode1* := *newMode*

SwitchFireControl2AutomaticMode $\hat{=}$ *mode1* := *automatic*

SwitchFireControl2DisabledMode $\hat{=}$ *mode1* := *disabled*

FireSysStart $\hat{=}$
 systemState! *fireSysStarts* \rightarrow *switchOn* \rightarrow
 switched \rightarrow *InitFireControl* ; *FireSys*

FireSys $\hat{=}$
 systemState! *fireSys* \rightarrow
 (
 modeSwitch? *newMode* \rightarrow (*SwitchFireControlMode*(*newMode*) ;
 FireSys)
 □
 (*detection*? *newZone* \rightarrow *switched* \rightarrow *alarm*! *firstStage* \rightarrow
 if(*mode1* = *manual*) \rightarrow *Manual*
 ||(*mode1* = *automatic*) \rightarrow *Auto*
 fi
)
 □
 actuatorsReplaced \rightarrow *switched* \rightarrow
 (*SwitchFireControl2AutomaticMode* ; *Reset*)
 □
 (*fault*? *faultId* \rightarrow *switched* \rightarrow *FireSys*)
)

$Manual \triangleq systemState!manuals \rightarrow ((detection?newZone \rightarrow switched \rightarrow Manual) \sqcap (silenceAlarm \rightarrow Manual))$

$Auto \triangleq systemState!autos \rightarrow ((reset \rightarrow (switched \rightarrow InitFireControl ; FireSys)) \sqcap (detection?newZone \rightarrow Auto))$

$Reset \triangleq systemState!resets \rightarrow ((detection?newZone \rightarrow switched \rightarrow Reset) \sqcap (reset \rightarrow switched \rightarrow Reset))$

$WaitingClock \triangleq (detection?newZone \rightarrow switched \rightarrow WaitingClock) \sqcap (clockFinished \rightarrow Discharge)$

$Countdown \triangleq systemState!countdowns \rightarrow startClock \rightarrow WaitingClock$

$FireSysD \triangleq systemState!fireSysDs \rightarrow ((detection?newZone \rightarrow switched \rightarrow alarm!firstStage \rightarrow FireSysD) \sqcap (alarm!firstStage \rightarrow FireSysD))$

$Discharge \triangleq systemState!discharges \rightarrow automaticDischarge \rightarrow \mathbf{var} \ log : \mathbb{P} \ AreaId \bullet ((log := \emptyset) ; \mathbf{while} \ (log \neq \emptyset) \ \mathbf{do} \ log := log \cup \{newZone\} \ \mathbf{end})$

$Disabled \triangleq systemState!disableds \rightarrow ((detection?newZone \rightarrow switched \rightarrow Disabled) \sqcap (silenceAlarm \rightarrow Disabled))$

$\bullet \ FireSysStart$

end

process $Area \triangleq \ ident : AreaId \bullet \mathbf{begin}$

state $AreaState \ == \ [\ mode : Mode ; \ controlledZones : \mathbb{P} \ ZoneId ; \ activeZones : \mathbb{P} \ ZoneId ; \ discharge : \mathbb{P} \ ZoneId ; \ disabled : \mathbb{P} \ ZoneId ; \]$

$\overline{\text{controlledZones}} = getZones \ ident \wedge (mode = automatic) \Rightarrow active = ttrue \Leftrightarrow \#activeZones \geq 2$

$InitArea \triangleq activeZones := \emptyset ; discharge := \mathbf{ffalse} ; mode := automatic$

$SwitchAreaMode \triangleq newMode : Mode \bullet mode := newMode$

$SwitchArea2AutomaticMode \triangleq mode := automatic$

$SwitchArea2DisabledMode \triangleq mode := disabled$

$ActivateZone \triangleq newZone : ZoneId \bullet (\text{if } newZone \in controlledZones \rightarrow activeZones := ($

$AutomaticDischarge \triangleq discharge := active$

$ManualDischarge \triangleq pAreas : \mathbb{P} AreaId \bullet (\text{if } ident \in pAreas \wedge active = ttrue \rightarrow discharge$

$StartArea \triangleq switchOn \rightarrow InitArea ; AreaCycle$

$AreaCycle \triangleq (actuatorsReplaced \rightarrow (SwitchArea2AutomaticMode ; ResetArea)) \sqcap (mod$

$AutoArea \triangleq \text{if}(active = ttrue) \rightarrow countdown \rightarrow counting \rightarrow WaitingDischarge \parallel (active$

$ManualArea \triangleq (silenceAlarm \rightarrow ResetArea) \sqcap (detection?newZone \rightarrow (ActivateZone(n$

$WaitingDischarge \triangleq (detection?newZone \rightarrow WaitingDischarge) \sqcap (automaticDischarge$

$ResetArea \triangleq (detection?newZone \rightarrow (ActivateZone(newZone) ; ResetArea)) \sqcap (reset \rightarrow$

$AreaD \triangleq (detection?newZone \rightarrow (ActivateZone(newZone) ; DisabledArea)) \sqcap (actuator$

$DisabledArea \triangleq (detection?newZone \rightarrow (ActivateZone(newZone) ; DisabledArea)) \sqcap (si$

$\bullet StartArea$

end

process $ConcreteAreas \triangleq \parallel ident : AreaId \parallel [Sigmaareas] \bullet Area(ident)$

process $DisplayController \triangleq \text{begin}$

$StartDisplay \triangleq switchOn \rightarrow switchLamp.systemOnLamp!on \rightarrow switched \rightarrow DisplayCyc$

As the strategy does not accept iterated interleaving, I have to expand this for all the types. AreaId ::= A0 — A1 ZoneId ::= Z0 — Z1 — Z2 — Z3 — Z4 — Z5 LampId ::= zoneFaultLamp — earthFaultLamp — sounderLineFaultLamp — powerFaultLamp — isolateRemoteSignalLamp — actuatorLineFaultLamp — circuitFaultLamp — alarmSilencedLamp — systemOnLamp {circuitFaultLamp,systemOnLamp}

$SwitchLampsOff \triangleq (((switchBuzzer!off \rightarrow Skip) \parallel \{\} \parallel \{\}) \parallel (switchLamp.zoneFaultLamp!off \rightarrow$

$DisplayCycle \triangleq (detection?zone \rightarrow switchLamp.zone!on \rightarrow switched \rightarrow DisplayCycle) \square (fault?$

• *StartDisplay*

end

process *ConcreteInternalSystem* $\triangleq (FireControl \parallel [Sigma1] DisplayController) \setminus DisplaySync$