Safety-Critical Java in *Circus*

University of York

15 November, 2011

Overview

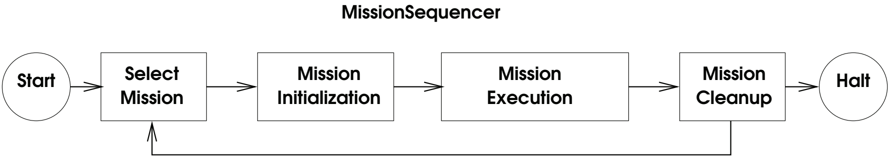
Safety-Critical Java

- Nothing about design techniques
- Level 1

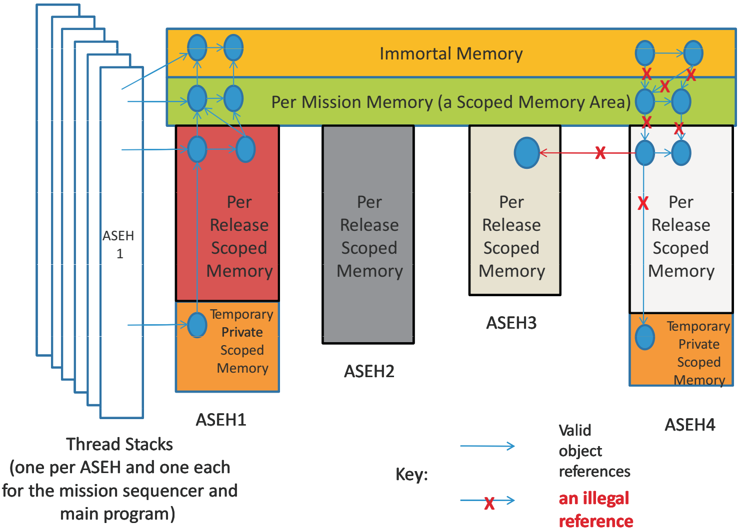
Refinement technique for SCJ

- Timing requirements and their decomposition
- Value-based specification and class-based designs
- SCJ memory model

Application structure



Scoped memory area



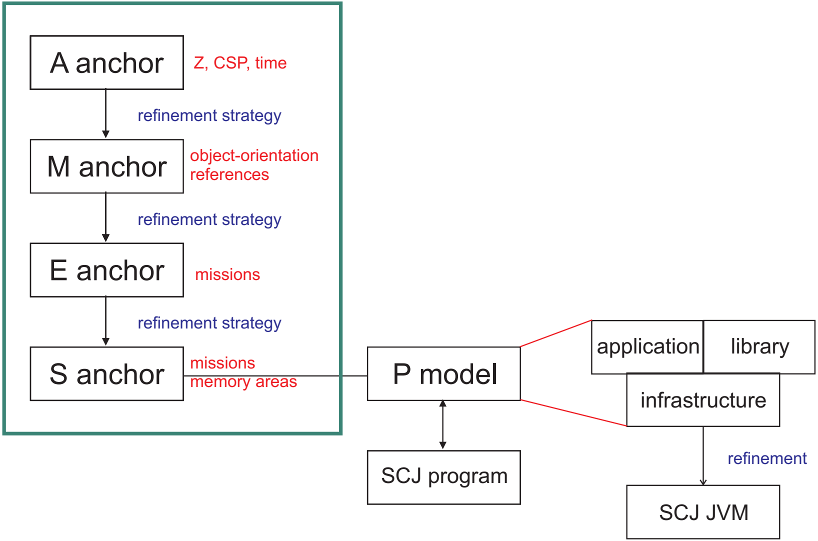
Circus Family

- *Circus*: Z + CSP + ZRC
- Language for refinement
- Target programming languages: occam, Handel-C, SPARK Ada
- Processes: encapsulate state + behaviour
  - State: Z
  - Actions: CSP + Z + guarded command language
  - Communication: through channels
- Semantic model: Unifying Theories of Programming

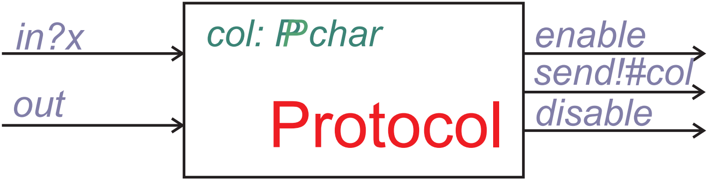
Circus variants

- *Circus Time*
- *OhCircus*
- ...

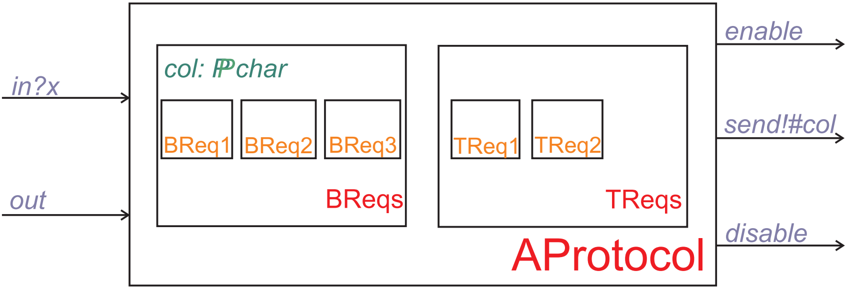
Development of SCJ programs



Example: simple protocol



Example: A Anchor



Example: A Anchor

process $BReqs \mathrel{\widehat=}$ **begin**

state $APState == [col : \mathbb{P}\,\mathrm{char}]$

$Init == [APState' \mid col' = \varnothing]$

$Insert == [\Delta APState;\ x? : \mathrm{char} \mid col' = col \cup \{x?\}]$

$InsS(w) \mathrel{\widehat=} (\mathbf{wait}\ 0..w\ ;\ Insert)\ \Box\ (send!(\#\, col)@t \longrightarrow InsS(w - t))$

$BReq1 \mathrel{\widehat=} (in?x@t \longrightarrow InsS(100 - t)\ \Box\ send!(\#\, col) \longrightarrow \mathbf{Skip})\ ;\ BReq1$

$BReq2 \mathrel{\widehat=} out \longrightarrow enable \longrightarrow send?x \longrightarrow BReq2$

$BReq3 \mathrel{\widehat=} send?x \longrightarrow disable \longrightarrow BReq3$

$\bullet\quad \mathbf{wait}\ 0\,..\,3\ ;\ Init;$
$(\ BReq1\ \ [\![\ \{col\} \mid \{\!|\ send\ |\!\} \mid \{\}\ ]\!]\ \ (\ BReq2\ \ [\![\ \{\!|\ send\ |\!\}\ ]\!]\ \ BReq3\ )\ )$

end

Example: A Anchor

process *TReqs* $\widehat{=}$ **begin**

$TReq1 \mathrel{\widehat{=}} ((in?x \longrightarrow \mathbf{Skip}) \blacktriangleright 5 \interleave \mathbf{wait}\ 100)\ ;\ \ TReq1$

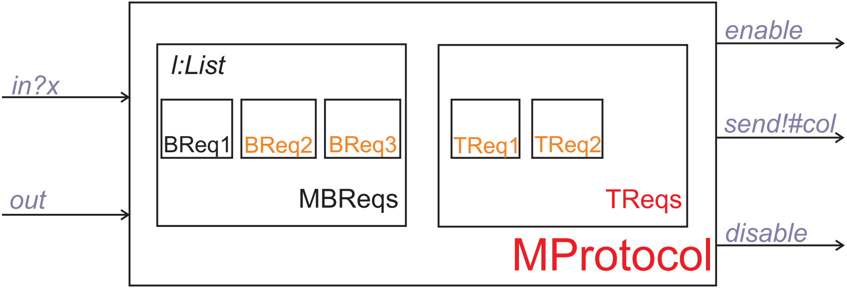
$TReq2 \mathrel{\widehat{=}}\ \ out \longrightarrow \mathbf{wait}\ 0 \ldots 7\ ;\ \ enable \longrightarrow (disable \longrightarrow \mathbf{Skip}) \blacktriangleright 15;$
$\quad TReq2$

$\bullet\ \ TReq1 \interleave TReq2$

end

system $AProtocol \mathrel{\widehat{=}} BReqs \llbracket \{\!|\ in, out, enable, disable\ |\!\} \rrbracket\ TReqs$

Example: M Anchor



M **Anchor**

Memory allocation

- Java memory model
- Language: *OhCircus* with references
- Data refinement
- Automation: not possible in general

Example: M Anchor

class *List* $\widehat{=}$

state *LState* $==$ [*val* : char; *next* : *List*; *empty* : *Bool* | . . .]

initial *Init* $==$ [*LState*$'$ | *empty*$'$ = *true*]

synchronized public *insert*

Δ*LState*; $x?$: char

let *col* $==$ **self**.*elems*(); *col*$'$ $==$ **self**$'$.*elems*() $\bullet$ *col*$'$ $=$ *col* $\cup \{x?\}$

logical *elems* $\widehat{=}$ **res** *col* : $\mathbb{P}$ char $\bullet$

if *empty* $=$ *true* $\longrightarrow$ *col* $:= \varnothing$

$[\!]$ *empty* $=$ *false* $\longrightarrow$ *col* $:=$ *next*.*elems*() $\cup \{$*val*$\}$

fi

synchronized public *size* $==$ [Ξ*LState*; $s!$: $\mathbb{Z}$ | . . .]

end

Example: M Anchor

$\textbf{process}\ \ MBReqs \mathrel{\widehat=} \textbf{begin}$

$\textbf{state}\ \ MPState == [l : List]$

$Init \mathrel{\widehat=} (l := \textbf{new}\ List)$

$InsS(w) \mathrel{\widehat=} (\textbf{wait}\ 0..w\ ;\ \ l.insert(x))\ \ \Box\ \ (send!(l.size())@t \longrightarrow InsS(w - t))$

$BReq1 \mathrel{\widehat=} (in?x@t \longrightarrow InsS(100 - t)\ \ \Box\ \ send!(l.size()) \longrightarrow \textbf{Skip})\ ;\ \ BReq1$

$\ldots$

E **Anchor**

Design of missions and handlers

- Language: no change
- Four phases of refinement
  - CP: collapse parallelism
  - SH: sharing
  - MH: missions and handlers
  - AR: algorithmic refinement
- Automation
  - Interface of the handlers?
  - Sharing among handlers?

Example: CP phase

Example: CP phase

$\textbf{system}\ \mathit{EProtocol} \mathrel{\widehat{=}} \textbf{begin}$

$\textbf{state}\ \ \mathit{MPState} == [l : \mathit{List}]$

$\mathit{Init} \mathrel{\widehat{=}} (l := \textbf{new}\ \mathit{List})$

$$
\begin{array}{ll}
\mathit{InPending}(t, d) \mathrel{\widehat{=}} & (in?x@u \longrightarrow \mathit{AfterInPinsert}(t + u, 100 - (t + u), x)) \blacktriangleleft d \\
& \Box \\
& out@u \longrightarrow \mathit{InAfterOut}(t + u, d - u, 7)
\end{array}
$$

$\mathit{AfterInPinsert}(t, wins, x) \mathrel{\widehat{=}}$

$\quad \bigsqcap d : 0 \mathrel{..} wins \bullet (out@u \longrightarrow \ldots$

$\ldots$

$\bullet\ \ \textbf{wait}\ 0 \mathrel{..} 3\ ;\ \ \mathit{Init}\ ;\ \ \mathit{InPending}(0, 5)$

$\textbf{end}$

Example: CP phase

$\mathbf{system}\ \mathit{EProtocol} \mathrel{\widehat=} \mathbf{begin}$

$\mathbf{state}\ \ \mathit{MPState} == [l : \mathit{List}]$

$\mathit{Init} \mathrel{\widehat=} (l := \mathbf{new}\ \mathit{List})$

$$
\begin{array}{ll}
\mathit{InPending}(t, d) \mathrel{\widehat=} & (in?x@u \longrightarrow \mathit{AfterInPinsert}(t + u, 100 - (t + u), x)) \blacktriangleleft d \\
& \Box \\
& out@u \longrightarrow \mathit{InAfterOut}(t + u, d - u, 7)
\end{array}
$$

$\mathit{AfterInPinsert}(t, wins, x) \mathrel{\widehat=}$

$\quad \bigsqcap d : 0 \mathrel{..} wins \bullet (out@u \longrightarrow \ldots$

$\ldots$

$\bullet\ \ \mathbf{wait}\ 0 \mathrel{..} 3\ ; \ \boxed{\mathit{Init}\ ;\ \ \mathit{InPending}(0, 5)}$

$\mathbf{end}$

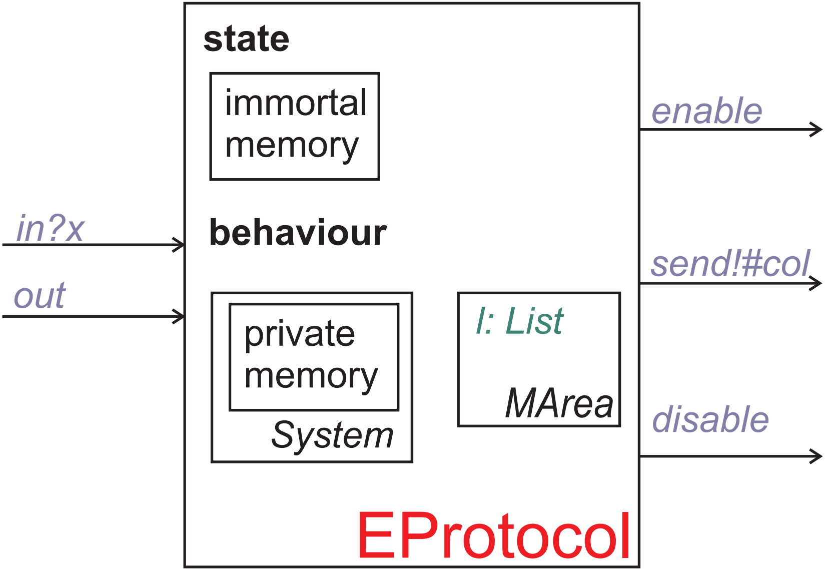
SH **Phase**

Splitting the state

Components in

- Immortal memory: stay where they are
- Per-release and temporary areas: become local to the main action
- Mission memory: become local to a new separate parallel action

Example: SH phase



Example: SH phase

$$\textbf{system}\ \mathit{EProtocol} \mathrel{\widehat=} \textbf{begin}$$

$\ldots$

$InPending(t, d) \mathrel{\widehat=} \ldots$

$\ldots$

$System \mathrel{\widehat=} InPending(0, 5)$

$MArea \mathrel{\widehat=}$

$$\left(\begin{array}{l} \textbf{var}\ l : List \bullet Init; \\ \left(\mu\ X \bullet \left(\begin{array}{l} insertLC?x \longrightarrow l.insert(x)\ ;\ \ insertLR \longrightarrow X \\ \Box \\ sizeLC \longrightarrow sizeLR!(l.size(x)) \longrightarrow X \end{array}\right)\right) \end{array}\right)$$

$\bullet$ $\textbf{wait}\ 0 \mathrel{..} 3;$
$(System \llbracket\ \{\!|\ insertLC, insertLR, \ldots\ |\!\}\ \rrbracket\ MArea) \setminus \{\!|\ insertLC, insertLR, \ldots\ |\!\}$

end

Example: SH phase

system *EProtocol* $\widehat{=}$ **begin**

$\ldots$

$InPending(t, d) \widehat{=} \ldots$

$\ldots$

$$System \widehat{=} \boxed{InPending(0, 5)}$$

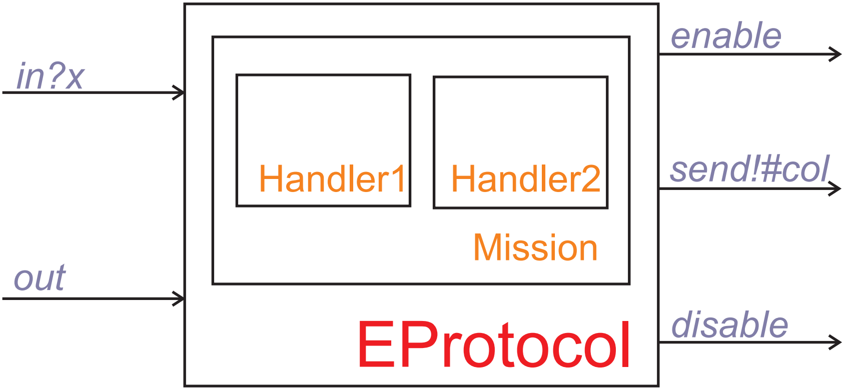
$$MArea \widehat{=}$$

$$\left(\begin{array}{l} \mathbf{var}\ l : List \bullet Init; \\ \quad \left(\mu\ X \bullet \left(\begin{array}{l} insertLC?x \longrightarrow l.insert(x)\ ;\ insertLR \longrightarrow X \\ \Box \\ sizeLC \longrightarrow sizeLR!(l.size(x)) \longrightarrow X \end{array}\right)\right) \end{array}\right)$$

$$\bullet\ \mathbf{wait}\ 0\,..\,3;$$
$$(System \llbracket \{\!|\ insertLC, insertLR, \ldots\ |\!\} \rrbracket\ MArea) \setminus \{\!|\ insertLC, insertLR, \ldots\ |\!\}$$

end

Example: MH phase



Example: MH phase

$\textbf{system}\ \mathit{EProtocol} \mathrel{\widehat=} \textbf{begin}$

$\ldots$

$\mathit{Handler1} \mathrel{\widehat=}$

$\quad ((\mathit{in}?x@t \longrightarrow \textbf{wait}\ 0..(100 - t)\ ;\ \mathit{insertLC}!x \ldots$

$\mathit{Handler2} \mathrel{\widehat=}$

$\quad \mathit{out} \longrightarrow \mathit{sizeLC} \longrightarrow \mathit{sizeLR}?x \longrightarrow \textbf{wait}\ 0 \mathrel{..} 7;$

$\quad \mathit{enable} \longrightarrow (\mathit{send}!x \longrightarrow \mathit{disable} \longrightarrow \textbf{Skip}) \blacktriangleright 15\ ;\ \mathit{Handler2}$

$\mathit{Mission} \mathrel{\widehat=} (\mathit{Handler1} \interleave \mathit{Handler2})$

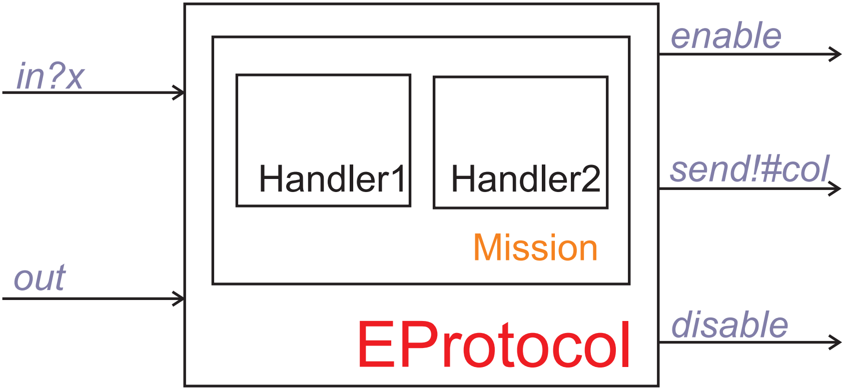
$\mathit{System} \mathrel{\widehat=} \mathit{Mission}$

$\mathit{MArea} \mathrel{\widehat=} \ldots$

- $\textbf{wait}\ 0 \mathrel{..} 3;$
  $(\mathit{System} \llbracket \{\!| \mathit{insertLC}, \mathit{insertLR}, \ldots |\!\} \rrbracket \mathit{MArea}) \setminus \{\!| \mathit{insertLC}, \mathit{insertLR}, \ldots |\!\}$

$\textbf{end}$

Example: AR phase



S **Anchor**

SCJ framework

- Language: *SCJ-Circus*
- Abbreviations
- Underlying: same language + SCJ memory model
- Refinement laws for new constructs

Example: S Anchor

$$
\begin{array}{l}
\textbf{sequencer}\ \mathit{MainMissionSequencer} \mathrel{\widehat=} \textbf{begin}\\
\textbf{state}\ \ \mathit{MainMissionSequencerState} == [\mathit{mission\_done} : \mathit{Bool}]\\
\textbf{initial} \mathrel{\widehat=} \mathit{mission\_done} := \mathit{false}\\
\textbf{getNextMission} \mathrel{\widehat=}\\
\quad \textbf{if}\ \ \mathit{mission\_done} = \mathit{false} \longrightarrow\\
\qquad \mathit{mission\_done} := \mathit{true};\ \textbf{ret} := \mathit{ProtocolMission}\\
\quad [\!]\ \mathit{mission\_done} = \mathit{true} \longrightarrow \textbf{ret} := \mathit{null}\\
\quad \textbf{fi}\\
\textbf{end}
\end{array}
$$

$$
\begin{array}{l}
\textbf{mission}\ \ \mathit{ProtocolMission} \mathrel{\widehat=} \textbf{begin}\\
\textbf{state}\ \ \mathit{MState} == [l : \mathit{List}]\\
\textbf{initialize} \mathrel{\widehat=}\\
\quad l := \textbf{new}\,\mathit{List}\ ;\ \ (\textbf{newHandler}\,\mathit{Handler1}(l))\ ;\ \ (\textbf{newHandler}\,\mathit{Handler2}(l))\\
\textbf{cleanup} \mathrel{\widehat=} \textbf{Skip}\\
\textbf{end}
\end{array}
$$

Example: S Anchor

$\mathbf{periodic}(100)$ **handler** $\mathit{Handler1} \mathrel{\widehat=}$ **begin**
state $\mathit{Handler1\_State} == [l : \mathit{List}]$
initial $\mathit{Handler1\_Init} \mathrel{\widehat=} \mathbf{val}\ \mathit{list?} : \mathit{List} \bullet l := \mathit{list?}$
$\mathbf{handleAsyncEvent}(x, w) \mathrel{\widehat=} \mathbf{wait}\ 0..w \ ;\ l.\mathit{insert}(x)$
$\mathbf{dispatch} \mathrel{\widehat=} (\mathit{in?x@t} \longrightarrow \mathbf{handleAsyncEvent}(x, 100 - t)) \blacktriangleleft 5$
end

aperiodic handler $\mathit{Handler2} \mathrel{\widehat=}$ **begin**
state $\mathit{Handler2\_State} == [l : \mathit{List}]$
initial $\mathit{Handler2\_Init} \mathrel{\widehat=} \mathbf{val}\ \mathit{list?} : \mathit{List} \bullet l := \mathit{list?}$
$\mathbf{handleAsyncEvent} \mathrel{\widehat=}$
$\quad \mathbf{var}\ \mathit{size} : \mathbb{N} \bullet \mathit{size} := l.\mathit{size}()\ ;\ \mathbf{wait}\ 0\mathrel{..}7;$
$\quad\quad \mathit{enable} \longrightarrow (\mathit{send}\,!\,\mathit{size} \longrightarrow \mathit{disable} \longrightarrow \mathit{Skip}) \blacktriangleright 15$
$\mathbf{dispatch} \mathrel{\widehat=} (\mathit{out} \longrightarrow \mathbf{handleAsyncEvent}())$
end

S **Anchor: applications**

- Use *Circus* and the UTP for reasoning
- Automatic generation of SCJ programs
- Conversely: automatic generation of S models
  - Programming patterns
  - Refactoring
  - Examples?
- Identification of good programming practices?
- Basis to verify an SCJ implementation

Challenges ahead

Theory

- Integration of languages and theories
- Mechanisation
- Refinement laws and detailed strategies
- Modular reasoning about libraries

Practice

- Case studies
- Design patterns
- Generation of abstract models
- Automation

And beyond

- Certification, Resources, ...