1 Introduction

```
%------------------------------------------------------------------------------
%-- A REACTIVE BUFFER    - Case Study                                         -
%-- Action Refinement: controller and ring partitions                         -
%-- Example extracted from the paper "A Refinement Strategy for Circus"       -
%------------------------------------------------------------------------------
```

You must always include the circus_toolkit, and also check inside it to see the LaTeX commands the parser recognises (or to create your own commands)

2 Channels

Section name is to be the same as the name of the file, is possible. It helps IDEs

$\mathbf{section}\ \mathit{buffer\_refinement\_multienv}\ \mathbf{parents}\ \mathit{circus\_toolkit}$

Z paragraphs can be defined at top level

$\mathit{maxbuff}, \mathit{maxring} : \mathbb{N}$

CSP paragraphs can be defined at top-level

$\mathbf{channel}\ \mathit{input}, \mathit{output} : \mathbb{N}$

$\mathbf{channel}\ \mathit{write}, \mathit{read} : (1 \mathrel{..} \mathit{maxring}) \times \mathbb{N}$

$\mathbf{channel}\ \mathit{read\_1} : (1 \mathrel{..} \mathit{maxring})$

$\mathbf{channel}\ \mathit{read\_2} : \mathbb{N}$

With \CircusDeclSummary you can create a summary of declarations

3 Buffer

Unfortunately boxed processes (those spread across multiple begin/end Circus) are not yet available. You need to define your processes within one environment only. The only real problem is for axiomatic definitions (that I am looking into now). Schemas can be given horizontally.

A circus environment starts the process context

$\mathbf{process}\ \mathit{BufferMultiEnv} \mathrel{\widehat=} \mathbf{begin}$

Declarations	**This Section**	**Globally**
Unboxed items	5	5
Axiomatic definitions	1	1
Generic axiomatic defs.	0	0
Schemas	0	0
Generic schemas	0	0
Total	**6**	**6**

Table 1: Summary of *Circus* declarations for Section 2.

Z paragraphs within a process context are local to that process. They can be used within the process' actions but are unknown outside the process. That means names might be repeated, so long as they are globally unique, as well as locally unique (i.e. variable x can be declared globally, and within a process P and another Q with different types. You cannot declare a process named x, though, as this would duplicate the name x globally).

$$
\begin{array}{|l}
\textit{ControllerState} \\
\hline
\textit{cache} : \mathbb{N} \\
\textit{size} : 0 \mathinner{\ldotp\ldotp} \textit{maxbuff} \\
\textit{ringsize} : 0 \mathinner{\ldotp\ldotp} \textit{maxring} \\
\textit{top}, \textit{bot} : 1 \mathinner{\ldotp\ldotp} \textit{maxring} \\
\hline
(\textit{ringsize} \bmod \textit{maxring}) = ((\textit{top} - \textit{bot}) \bmod \textit{maxring}) \\
\textit{ringsize} = \textit{size} - 1 \\
\hline
\end{array}
$$

Schemas can be given in multiple horizontal paragraph within a single Z environment.

$$
\begin{array}{lcl}
\textit{RingState} & == & [\, \textit{ring} : \operatorname{seq} \mathbb{N} \mid \# \textit{ring} = \textit{maxring} \,] \\
\textit{CBufferState} & == & (\textit{ControllerState} \lor \textit{RingState})
\end{array}
$$

Circus state is marked accordingly within a circusaction environment

$$\textbf{state} \;\; \textit{CBufferState}$$

$$
\begin{array}{|l}
\textit{ControllerInit} \\
\hline
\textit{ControllerState}'; \; \textit{RingState}' \\
\hline
\textit{size}' = 0 \\
\textit{bot}' = 1 \\
\textit{top}' = 1 \\
\hline
\end{array}
$$

$$\begin{array}{|l}
\multicolumn{1}{l}{\mathit{CacheInput}} \\ \hline
\Delta \mathit{ControllerState} \\
\Xi \mathit{RingState} \\
x? : \mathbb{N} \\ \hline
(\mathit{size} = 0) \land (\mathit{size}' = 1) \\
(\mathit{cache}' = x?) \land (\mathit{bot}' = \mathit{bot}) \land (\mathit{top}' = \mathit{top}) \\ \hline
\end{array}$$

$$\begin{array}{|l}
\multicolumn{1}{l}{\mathit{StoreInput}} \\ \hline
\Delta \mathit{CBufferState} \\
x? : \mathbb{N} \\ \hline
(0 < \mathit{size}) \land (\mathit{size} < \mathit{maxbuff}) \\
(\mathit{size}' = \mathit{size} + 1) \land (\mathit{cache}' = \mathit{cache}) \\
(\mathit{bot}' = \mathit{bot}) \land (\mathit{top}' = (\mathit{top} \mathbin{\mathrm{mod}} \mathit{maxring}) + 1) \\
\mathit{ring}' = \mathit{ring} \oplus \{\mathit{top} \mapsto x?\} \\ \hline
\end{array}$$

$$\begin{array}{|l}
\multicolumn{1}{l}{\mathit{StoreInputController}} \\ \hline
\Delta \mathit{ControllerState} \\
\Xi \mathit{RingState} \\ \hline
(0 < \mathit{size}) \land (\mathit{size} < \mathit{maxbuff}) \\
(\mathit{size}' = \mathit{size} + 1) \land (\mathit{cache}' = \mathit{cache}) \\
(\mathit{bot}' = \mathit{bot}) \land (\mathit{top}' = (\mathit{top} \mathbin{\mathrm{mod}} \mathit{maxring}) + 1) \\ \hline
\end{array}$$

Actions, and other *Circus* declarations **must** use `\circdef` instead of `\defs` or `==` from Z. This is needed to avoid parsing ambiguities with the Z grammar. One can use tabulation and blocks (see .tex file).

Because guards are Z predicates, and predicates can be parenthesised, we need a different token for disambiguation as well. So, every guard **requires** a `\lcircguard pred \rcircguard \circguard A` and it typesets as $\big(pred\big) \mathbin{\&} A$. Other similar marker-tokens are used for disambiguation.

Like for variable names, channel names can accept ? or ! or $'$. So, hard space is needed to indicate this is input prefix: you must type `input~?x` instead of simply `input?x`, and similarly for other strokes. Restricting channel names not to have strokes (i.e. `input~?x?` for the input on channel named $x?$) is not straightforward (i.e. such names are in context because of the Z type rules for inputs), and it does not solve the problem anyway.

Moreover, also notice the extra parenthesis within the input prefixing. This is important in this context because of potential precedence confusion to the parser. They keep the variable x in scope for the *StoreInput* schema, say.

$$\begin{array}{l}
\mathit{InputController} \;\widehat{=} \\
\quad \big(\mathit{size} < \mathit{maxbuff}\big) \mathbin{\&} (\mathit{input}\,?x \longrightarrow \\
\qquad (((\mathit{size} = 0) \mathbin{\&} \big(\mathit{CacheInput}\big)) \qquad\qquad\qquad\qquad\qquad) \\
\qquad \Box \\
\qquad ((\mathit{size} > 0) \mathbin{\&} \mathit{write.top}\,!x \longrightarrow \big(\mathit{StoreInputController}\big)))
\end{array}$$

$$\begin{array}{l}
\mathit{CInput} \;\widehat{=} \\
\quad \big(\mathit{size} < \mathit{maxbuff}\big) \mathbin{\&} (\mathit{input}\,?x \longrightarrow \\
\qquad\qquad\qquad (((\mathit{size} = 0) \mathbin{\&} \big(\mathit{CacheInput}\big))) \\
\qquad\qquad\qquad \Box \\
\qquad\qquad\qquad ((\mathit{size} > 0) \mathbin{\&} \big(\mathit{StoreInput}\big)))
\end{array}$$

$$\begin{array}{|l}
\multicolumn{1}{l}{\mathit{NoNewCache}} \\
\hline
\Delta \mathit{ControllerState} \\
\Xi \mathit{RingState} \\
\hline
\mathit{size} = 1 \\
\mathit{size}' = 0 \land \mathit{cache}' = \mathit{cache} \\
\mathit{bot}' = \mathit{bot} \land \mathit{top}' = \mathit{top} \\
\hline
\end{array}$$

Function application within Z schemas (as in Standard Z) do require hard spaces or parenthesis (i.e. `f~x` or `f(x)`), otherwise the soft space is eaten during lexing (i.e. `f x` becomes `fx`, which leads to a type error).

$$\begin{array}{|l}
\multicolumn{1}{l}{\mathit{StoreNewCache}} \\
\hline
\Delta \mathit{CBufferState} \\
\hline
\mathit{size} > 1 \\
\mathit{size}' = \mathit{size} - 1 \land \mathit{cache}' = \mathit{ring}~\mathit{bot} \\
\mathit{bot}' = (\mathit{bot} \bmod \mathit{maxring}) + 1 \land \mathit{top}' = \mathit{top} \\
\mathit{ring}' = \mathit{ring} \\
\hline
\end{array}$$

$$\begin{array}{|l}
\multicolumn{1}{l}{\mathit{StoreNewCacheController}} \\
\hline
\Delta \mathit{ControllerState} \\
\Xi \mathit{RingState} \\
x? : \mathbb{N} \\
\hline
\mathit{size} > 1 \\
\mathit{size}' = \mathit{size} - 1 \land \mathit{cache}' = x? \\
\mathit{bot}' = (\mathit{bot} \bmod \mathit{maxring}) + 1 \land \mathit{top}' = \mathit{top} \\
\hline
\end{array}$$

New hard lines (`\\` and `\also`) are option after `\circdef`.

$$\begin{array}{l}
\mathit{OutputController} \ \widehat{=} \\
\quad \big(\mathit{size} > 0\big) \mathbin{\&} \mathit{output}\,!\mathit{cache} \longrightarrow \\
\qquad \big(\big(\mathit{size} > 1\big) \mathbin{\&} \mathit{read.bot}\,?x \longrightarrow \big(\mathit{StoreNewCacheController}\big)\big) \\
\qquad \Box \\
\qquad \big(\big(\mathit{size} = 1\big) \mathbin{\&} \big(\mathit{NoNewCache}\big)\big)
\end{array}$$

$$\begin{array}{l}
\mathit{COutput} \ \widehat{=} \\
\quad \big(\mathit{size} > 0\big) \mathbin{\&} \mathit{output}\,!\mathit{cache} \longrightarrow \\
\qquad \big(\big(\mathit{size} > 1\big) \mathbin{\&} \big(\mathit{StoreNewCache}\big)\big) \\
\qquad \Box \\
\qquad \big(\big(\mathit{size} = 1\big) \mathbin{\&} \big(\mathit{NoNewCache}\big)\big)
\end{array}$$

One should be careful with the precedences. See the file process.tex in the type checker tests directory for this, and then properly include the parenthesis. See the Z standard precedence table for Z, and FDR manual for CSP precedence. Sequential composition is not just normal semicolon (;). This creates too many conflicts with Z and we used a different unicode/LaTeX symbol (;). It typesets just like ; , however. Missing-parenthesis errors are the harder to find and the worst in error generation!

$$\mathit{ControllerAction} \ \widehat{=} \ \big(\mathit{ControllerInit}\big) \,;\, (\mu\ X \bullet ((\mathit{InputController} \mathbin{\Box} \mathit{OutputController}) \,;\, X))$$

$$\begin{array}{|l}
\hline
\multicolumn{1}{l}{\mathit{StoreRingCmd}} \\
\Xi \mathit{ControllerState} \\
\Delta \mathit{RingState} \\
i? : 1 \mathinner{\ldotp\ldotp} \mathit{maxring} \\
x? : \mathbb{N} \\
\hline
\mathit{ring}' = \mathit{ring} \oplus \{i? \mapsto x?\} \\
\hline
\end{array}$$

$$\begin{array}{l}
\mathit{StoreRing} \ \widehat{=} \ \mathit{write}\,?i\,?x \longrightarrow \big(\mathit{StoreRingCmd}\big) \\
\mathit{NewCacheRing} \ \widehat{=} \ \mathit{read}\,?i\,!(\mathit{ring}\ i) \longrightarrow \mathbf{Skip} \\
\mathit{RingAction} \ \widehat{=} \ \mu\ X \bullet ((\mathit{StoreRing} \mathbin{\Box} \mathit{NewCacheRing}) \,;\, X)
\end{array}$$

$$\begin{array}{l}
\bullet\ (\qquad \mathit{ControllerAction} \\
\quad [\![\{\mathit{size}, \mathit{ringsize}, \mathit{cache}, \mathit{top}, \mathit{bot}\} \mid \{\!|\ \mathit{write}, \mathit{read}\ |\!\} \mid \{\mathit{ring}\}]\!] \\
\qquad \mathit{RingAction}) \\
\qquad\qquad \setminus \{\!|\ \mathit{write}, \mathit{read}\ |\!\}
\end{array}$$

end

Declarations	**This Section**	**Globally**
Unboxed items	4	9
Axiomatic definitions	0	1
Generic axiomatic defs.	0	0
Schemas	9	9
Generic schemas	0	0
Total	**13**	**19**

Table 2: Summary of *Circus* declarations for Section 3.