J Circus 2.0: an extension of an automatic translator from Circus to Java

Samuel Lincoln Magalhães Barrocas¹
Marcel Vinícius Medeiros Oliveira¹

¹Universidade Federal do Rio Grande do Norte, Brazil

Communicating Process Architectures 2012
Motivation

Why translating formal specifications into code?

It is possible to reduce the occurrence of errors in the implementation of software systems from formal specifications...

Figure: Formal specification to code
Motivation

Goal

To extend JCircus by providing

- an optimized strategy to translate the Multi-synchronisation primitive;
- a strategy to translate communications with arbitrary field decorations (e.g. `c?x!y.10`);
- a strategy to translate alphabetized parallelism;
- the translation of deadlock-free GUI processes to interact with the generated processes;
- the integration of JCircus with a refinement tool, called CRefine;
The Circus language

Circus Syntax

- Z + CSP + Refinement Calculus + G.C. language;
- Circus Specification - formed by paragraphs:
  - Z paragraph,
  - Channel declaration paragraph,
  - Channel set declaration paragraph and
  - Process paragraph;
The Circus language

Roulette

process $Roulette \triangleq$

begin

$\bullet \mu X \bullet start \rightarrow \left( \begin{array}{c}
result.RED \rightarrow X \\
\cap result.BLACK \rightarrow X
\end{array} \right)$

end

**Figure**: The Roulette process
The Circus language

Croupier

\[
\begin{align*}
\text{process } & \text{Croupier } \equiv \\
\text{begin} & \\
\text{StartRoulette } \equiv & \text{start } \rightarrow \text{TakeBet} \\
\text{TakeBet } \equiv & \text{enter?id } \rightarrow \text{bet.id?b } \rightarrow \text{result?x} \\
& \left( \text{if } (x = b) \rightarrow (\text{pay.ident } \rightarrow \text{Skip}) \right) \\
& \left( (x \neq b) \rightarrow \text{Skip} \right) \\
& \text{fi} \\
& \sqcap \text{stopBet } \rightarrow \text{Skip} \\
\bullet & \mu X \bullet \text{StartRoulette; } X \\
\text{end} & \\
\text{process } & \text{Table } \equiv \text{Roulette } || \{ \{ \text{start, result } \} \} || \text{Croupier}
\end{align*}
\]

\textbf{Figure}: The Croupier process
JCSP concepts

- Java API for implementing CSP constructs;
- Processes are classes with the CSProcess interface;
- The behavior of each process is implemented in its run method;
- Parallelism: Parallel class;
- External choice: Alternative;
- Multi-Synchronisation: AltingBarrier;
JCSP limitations

- Concurrency model different from CSP’s concurrency model;
- Partial implementation of the Communication primitive: JCSP channels only communicate one value at most;
- Multi-synchronisation without communication;
- External choice without alting processes;
J Circus’ definition

Java application that translates Circus specifications into Java code;
JCircus’ modules

- Parser
- Typechecker
- Pre-processor
- Translator
Figure: J Circus’ initial screen
Extensions provided to JCircus

- Optimized Multi-synchronisation;
- Complex communications;
- Alphabetized parallelism;
- Deadlock-free GUI;
- Integration with CRefine;
Extensions to J Circus

Optimized Multi-synchronisation

- Protocol x Alting Barriers;
- Use of alting barriers;
Complex communications :: Definition

- Simple communication x Complex communication; 
  \(ch?x?y\) is a complex communication; 
  \(ch.5?x\) and \(ch!4!7\) are not complex communications;

- JCSP does not implement communications with more than one field
Extensions to JCircus

Complex communications :: Strategy

- Expansion of the possibilities of communication;
- Assignment of the values to the input variables;
- Inference of the next action based on the chosen communication (in the case of an external choice);
Complex communications :: Strategy

\[ \{0, 1\} \times \{0, 1, 2, 3\} \]

\[ c_1 \overset{x?y}{\rightarrow} A_1(x, y) \quad \square \quad c_2 \overset{x?y}{\rightarrow} A_2(x, y) \]

\[ \{0 \mapsto \{x \mapsto 0, \ y \mapsto 0\}, \ldots, 7 \mapsto \{x \mapsto 1, \ y \mapsto 3\}\} \]

\[ \{0 \mapsto 0, \ldots, 7 \mapsto 0, 8 \mapsto 1, \ldots, 15 \mapsto 1\} \]

**Figure**: Example
Extensions to J Circus

Alphabetized Parallelism

- Hoare’s parallelism x Alphabetized parallelism
- JCSP: implements Hoare’s parallelism;
- Circus: Parallelism is alphabetized;
- Challenge: To force interleaving between occurrences of common events;
Alphabetized Parallelism :: Translation

The strategy uses channel renaming, and consists of the following steps:

- Identification of the parallel branches;
- Construction of the synchronisation sets;
- Defining the renaming of each parallel branch;
- Renaming processing;
- Hiding the renaming from the interface;
Extensions to J Circus

Alphabetized Parallelism

\[
\begin{align*}
(a \rightarrow \text{SKIP}) & \parallel \{ a \} \parallel (a \rightarrow \text{SKIP}) \\
\begin{array}{c}
0 \\
0
\end{array} & \downarrow \\
\begin{array}{c}
0 \\
0
\end{array} & \downarrow \\
\begin{array}{c}
(a0 \rightarrow \text{SKIP}) \\
\square a1 \rightarrow \text{SKIP}
\end{array} & \parallel CS \parallel \begin{array}{c}
(a0 \rightarrow \text{SKIP}) \\
(a1 \rightarrow \text{SKIP)
\end{array}
\end{align*}
\]

\[CS = \{ a0, a1 \}\]

\textbf{Figure}: Example
For each process generated in JCSP, it is generated, also in JCSP, a GUI that interacts with it;

There was a strategy for generating a GUI in the original version of JCircus, but it was not deadlock-free;
Extensions to J Circus

Deadlock-free GUI

Figure: Example
CRefine is a tool that automates the application of laws to refine specifications;

The integration demanded the migration of JCircus to CZT’s newest parser of Circus;
Integration with CRefine

Figure: Example
Performance Analysis

Multi-synchronisation

<table>
<thead>
<tr>
<th>Number of Processes</th>
<th>Translation (ms)</th>
<th>Execution (ms)</th>
<th>Memory (KB)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Protocol</td>
<td>Barriers</td>
<td>Protocol</td>
</tr>
<tr>
<td>3</td>
<td>257</td>
<td>115</td>
<td>157,8</td>
</tr>
<tr>
<td>4</td>
<td>314</td>
<td>130</td>
<td>170,8</td>
</tr>
<tr>
<td>5</td>
<td>254</td>
<td>143</td>
<td>152,8</td>
</tr>
<tr>
<td>30</td>
<td>1764</td>
<td>344</td>
<td>307,3</td>
</tr>
<tr>
<td>100</td>
<td>7052</td>
<td>17949</td>
<td>548,6</td>
</tr>
</tbody>
</table>

Figure: Example
Complex Communications

Figure: Example
Alphabetized Parallelism

<table>
<thead>
<tr>
<th>Number of Processes</th>
<th>Translation (ms)</th>
<th>Execution (ms)</th>
<th>Memory (KB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>298</td>
<td>23</td>
<td>0.98</td>
</tr>
<tr>
<td>3</td>
<td>306</td>
<td>22</td>
<td>1.25</td>
</tr>
<tr>
<td>4</td>
<td>326</td>
<td>21</td>
<td>1.52</td>
</tr>
<tr>
<td>5</td>
<td>330</td>
<td>26</td>
<td>1.78</td>
</tr>
<tr>
<td>6</td>
<td>365</td>
<td>25</td>
<td>2.05</td>
</tr>
<tr>
<td>7</td>
<td>377</td>
<td>26</td>
<td>2.31</td>
</tr>
<tr>
<td>8</td>
<td>412</td>
<td>28</td>
<td>2.58</td>
</tr>
<tr>
<td>9</td>
<td>433</td>
<td>25</td>
<td>2.84</td>
</tr>
<tr>
<td>10</td>
<td>469</td>
<td>29</td>
<td>3.11</td>
</tr>
</tbody>
</table>

**Figure:** Example
Conclusions and Future Work

- Translation of Hiding;
- Translation of Schema types;
- Optimization of complex communications;
Questions?