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

Samuel Lincoln Magalhães Barrocas[1]
Marcel Vinícius Medeiros Oliveira[1]

[1] Universidade Federal do Rio Grande do Norte, Brazil

Communicating Process Architectures 2012

Outline

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

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;

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;

Roulette

$$\begin{array}{l}\textbf{process } Roulette \mathrel{\widehat=} \\ \textbf{begin} \\ \qquad \bullet\ \mu\, X \bullet start \rightarrow \left(\begin{array}{l} result.RED \rightarrow X \\ \sqcap\ result.BLACK \rightarrow X \end{array}\right) \\ \textbf{end}\end{array}$$

Figure : The Roulette process

Croupier

$$
\begin{array}{l}
\textbf{process}\ Croupier \mathrel{\widehat=} \\
\textbf{begin} \\
\quad StartRoulette \mathrel{\widehat=}\ start \rightarrow TakeBet \\
\quad TakeBet \mathrel{\widehat=}\ enter?id \rightarrow bet.id?b \rightarrow result?x \\
\qquad\qquad \left(\begin{array}{l} \textbf{if}\ (x = b) \rightarrow (pay.ident \rightarrow Skip) \\ [\!](x \neq b) \rightarrow Skip \\ \textbf{fi} \end{array} \right) \\
\qquad\quad \sqcap\ stopBet \rightarrow Skip \\
\bullet\ \mu\, X \bullet StartRoulette;\ X \\
\textbf{end} \\
\textbf{process}\ Table \mathrel{\widehat=}\ Roulette \,[\![\, \{\!|\ start, result\ |\!\} \,]\!]\, Croupier
\end{array}
$$

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;

JCircus' definition

Java application that translates Circus specifications into Java code;

JCircus' modules

- Parser
- Typechecker
- Pre-processor
- Translator

JCircus' initial screen

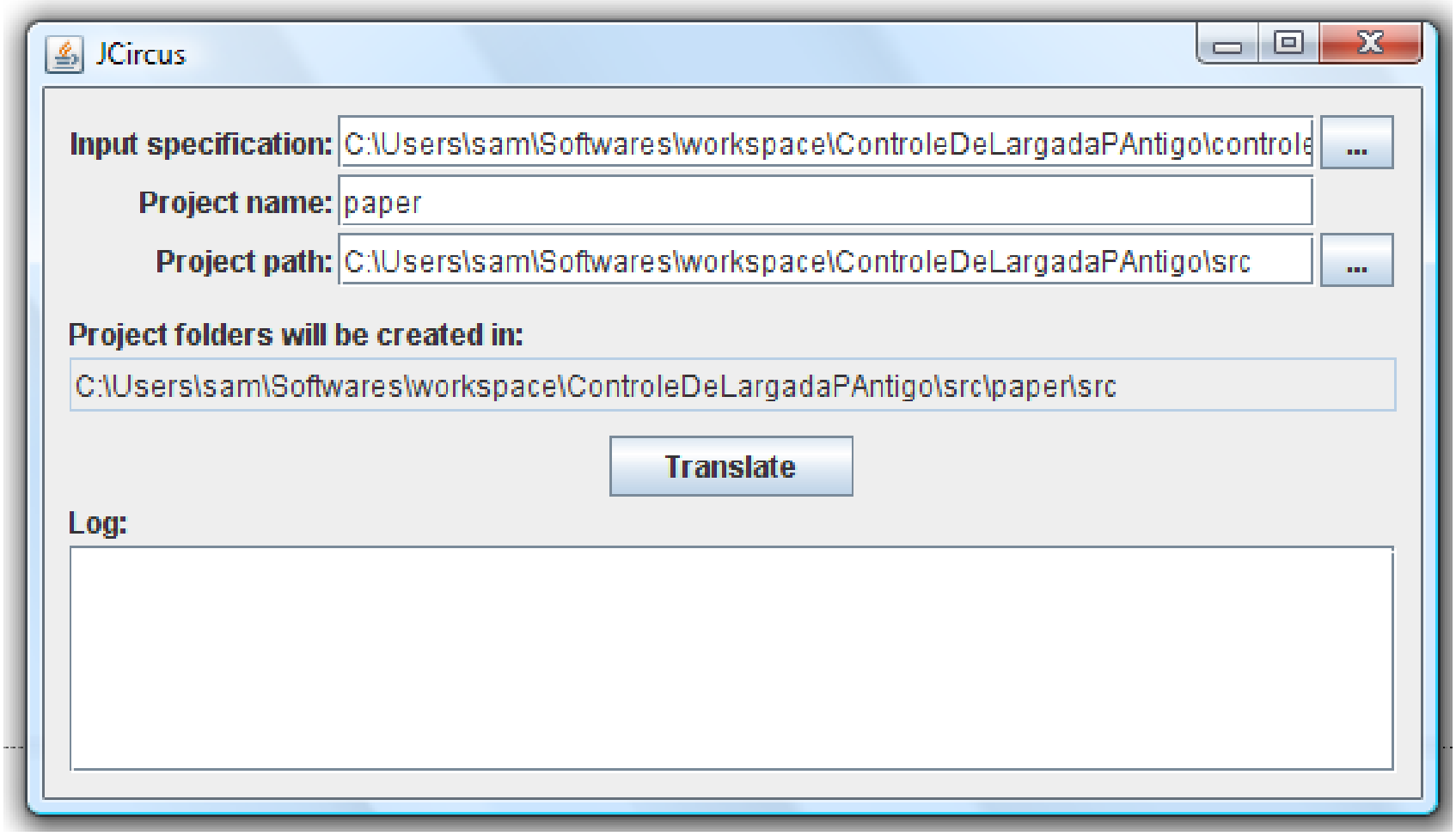


Figure : JCircus' initial screen

Extensions provided to JCircus

- Optimized Multi-synchronisation;
- Complex communications;
- Alphabetized parallelism;
- Deadlock-free GUI;
- Integration with CRefine;

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

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

$$\{0, 1\} \times \{0, 1, 2, 3\}$$

$$c_1?x?y \rightarrow A_1(x, y) \Box c_2?x?y \rightarrow A_2(x, y)$$

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

$$\{0 \mapsto 0, \ldots, 7 \mapsto 0, 8 \mapsto 1, \ldots, 15 \mapsto 1\}$$

Figure : Example

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;

Alphabetized Parallelism

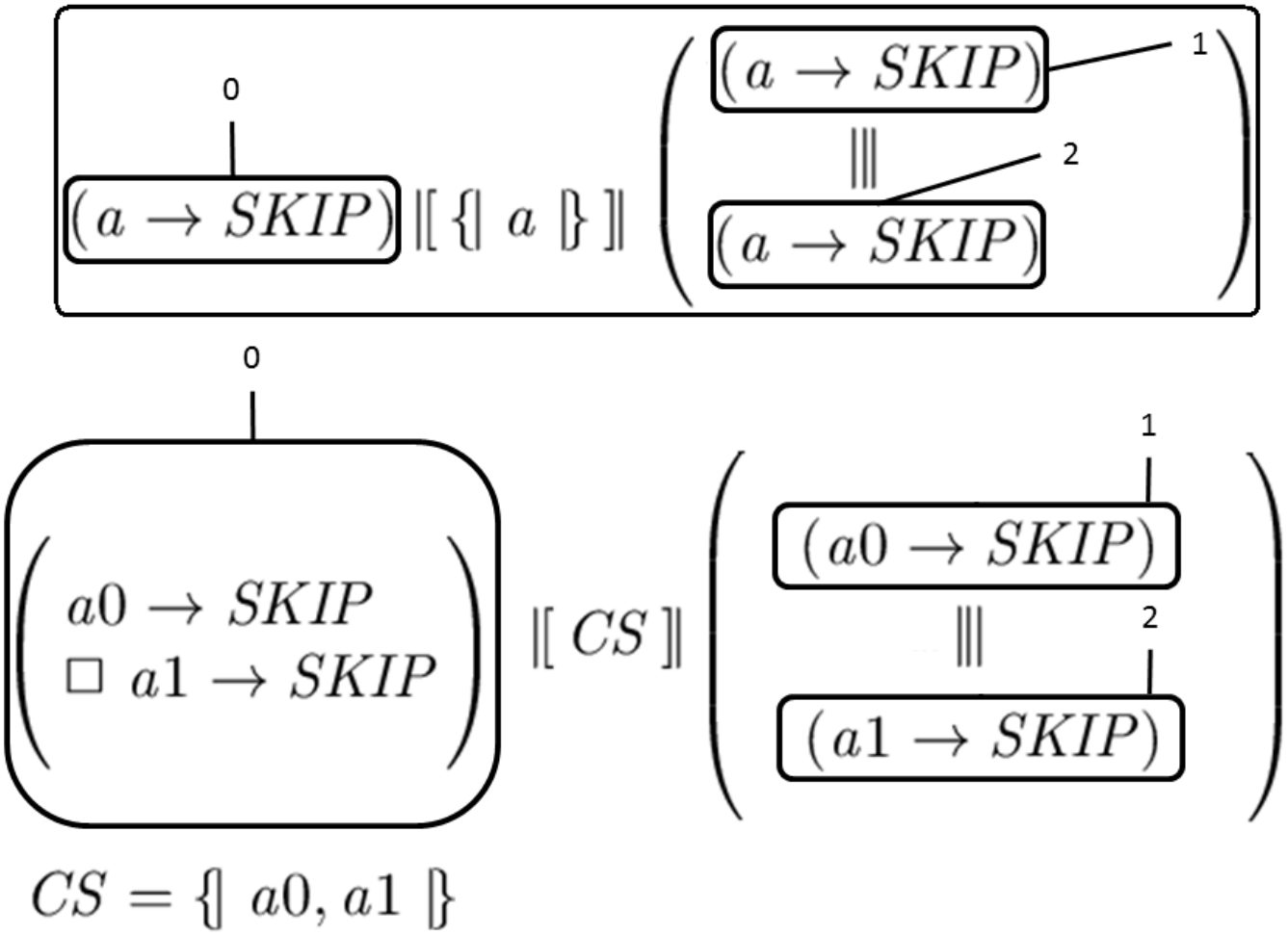


Figure : Example

Deadlock-free GUI

- 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;

Deadlock-free GUI

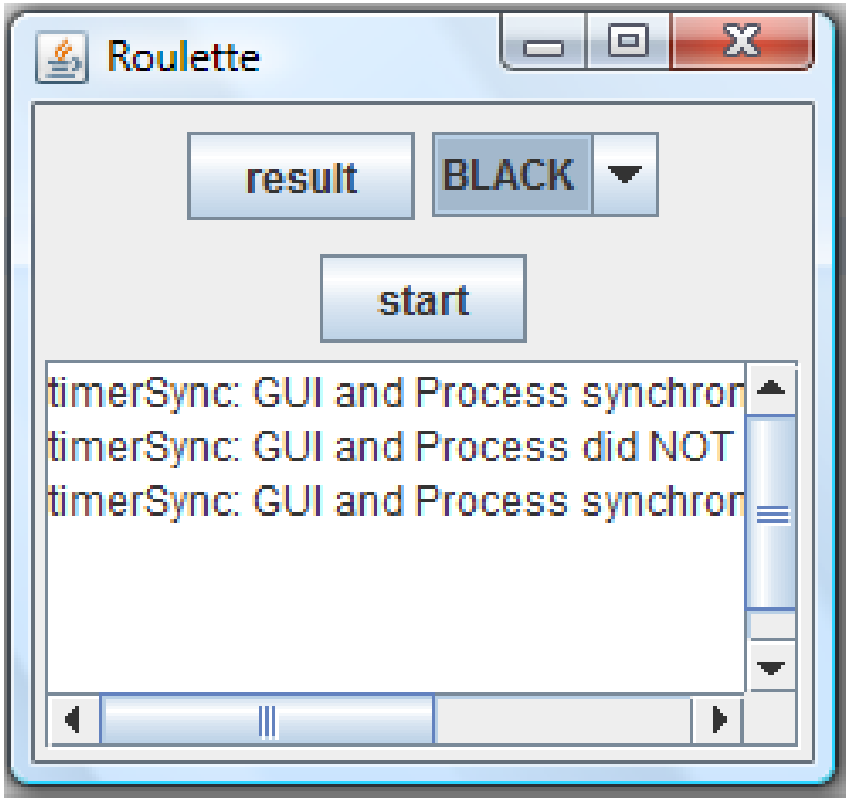


Figure : Example

Integration with CRefine

- 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

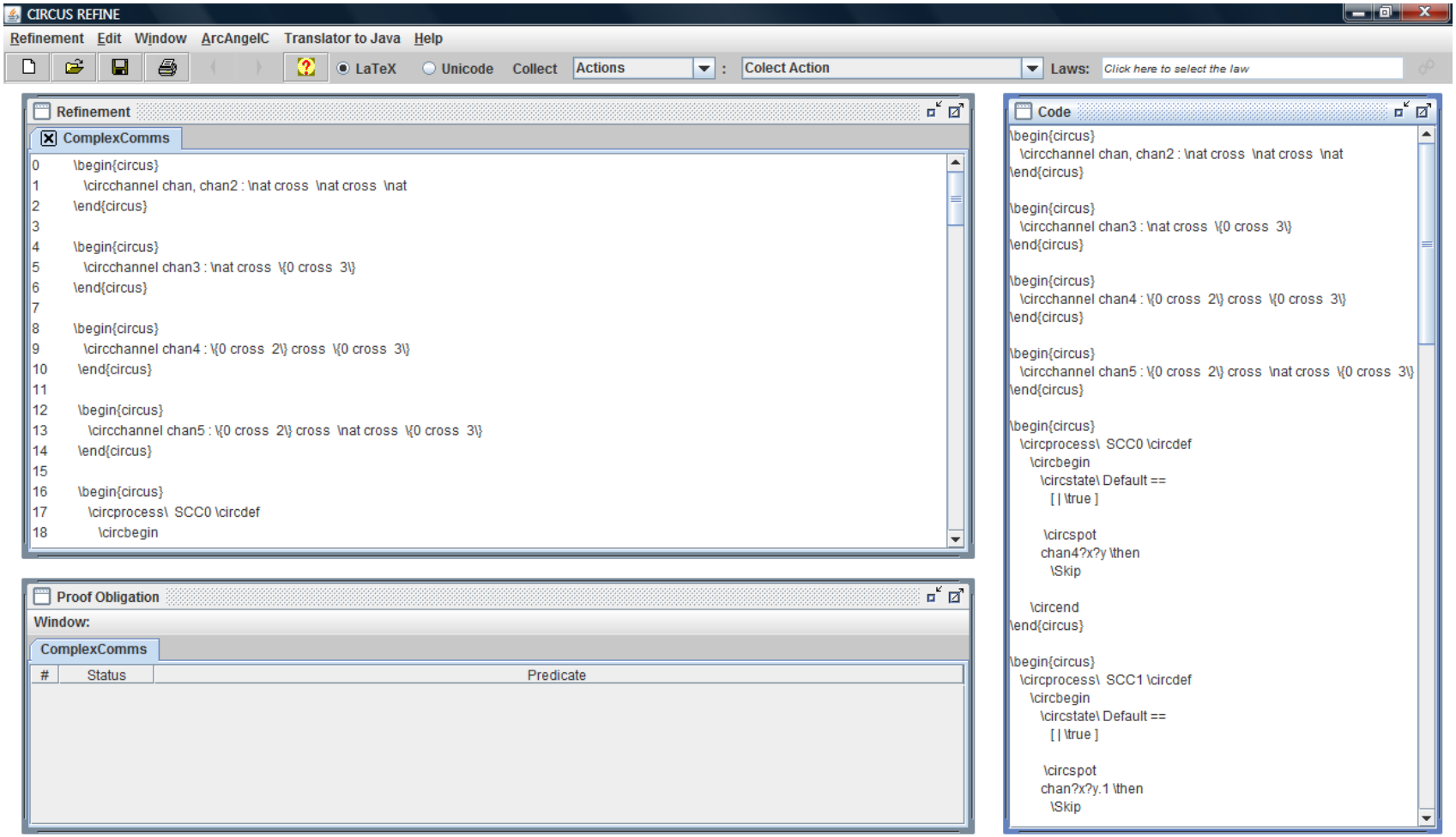


Figure : Example

Multi-synchronisation

Number of Processes	Translation (ms)		Execution (ms)		Memory (KB)	
	Protocol	Barriers	Protocol	Barriers	Protocol	Barriers
3	257	115	157,8	106,4	11276	14732
4	314	130	170,8	98	11232	14568
5	254	143	152,8	97,8	11208	14520
30	1764	344	307,3	136,6	11228	15636
100	7052	17949	548,6	378	11692	19680

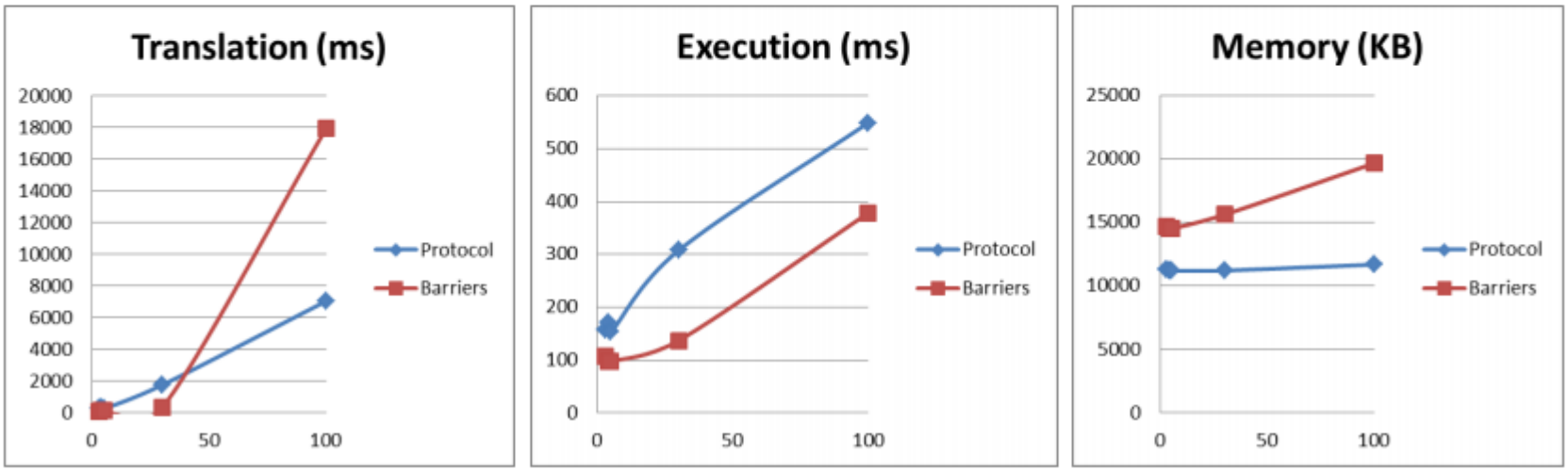
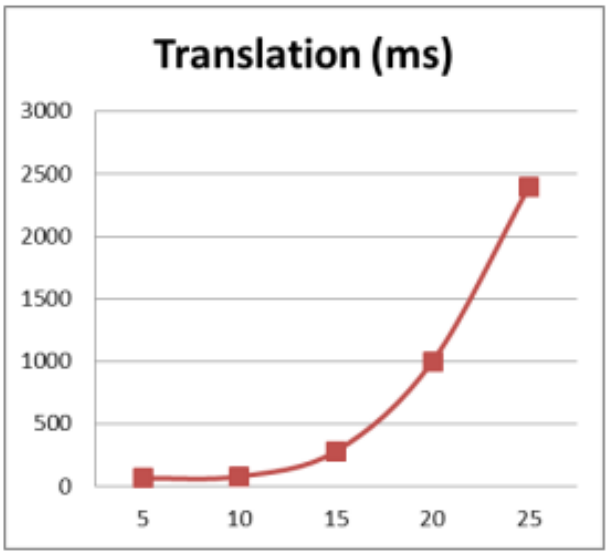


Figure : Example

Complex Communications

Type Size	Translation (ms)	Memory (KB)
5	68	0,78
10	83	0,94
15	285	1,09
20	1004	1,25
25	2403	1,38



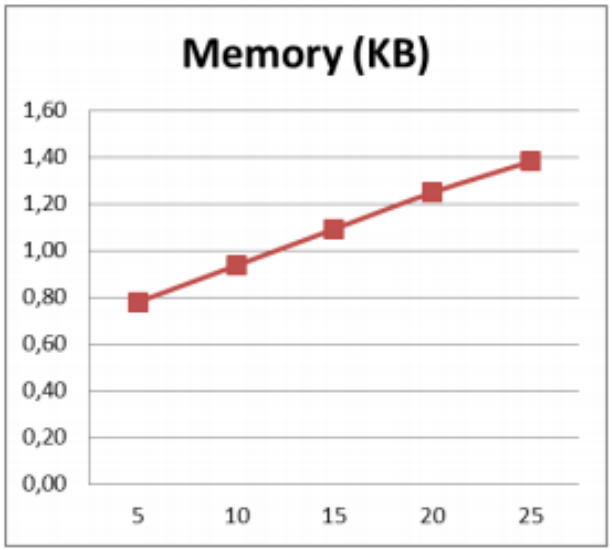
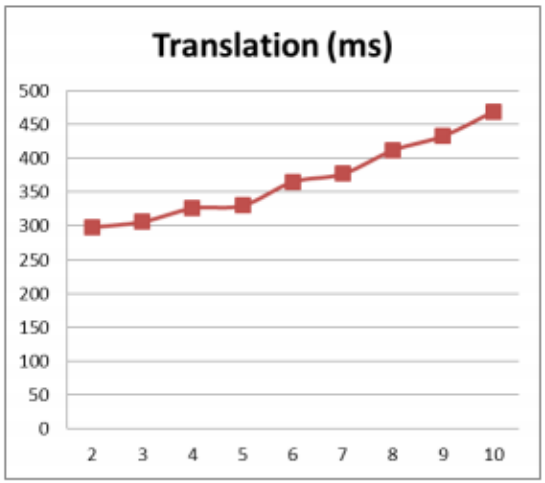


Figure : Example

Alphabetized Parallelism

Number of Processes	Translation (ms)	Execution (ms)	Memory (KB)
2	298	23	0,98
3	306	22	1,25
4	326	21	1,52
5	330	26	1,78
6	365	25	2,05
7	377	26	2,31
8	412	28	2,58
9	433	25	2,84
10	469	29	3,11



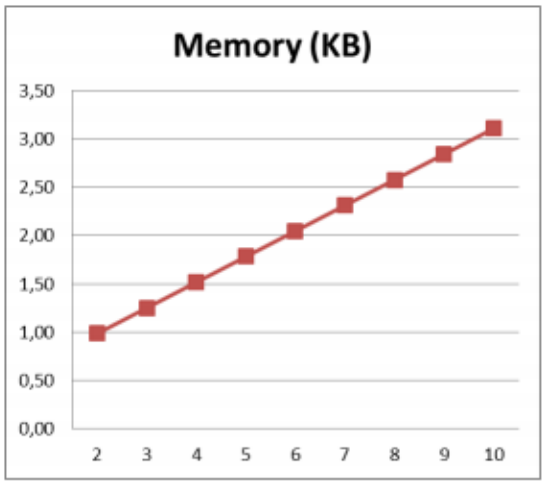


Figure : Example

Conclusions and Future Work

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

Questions?