Architectural Design Rewriting

R. Bruni\textsuperscript{1}, A. Lluch Lafuente\textsuperscript{1}, U. Montanari\textsuperscript{1}, E. Tuosto\textsuperscript{2}

Department of Computer Science, University of Pisa
\{bruni,lafuente,ugo\}@di.unipi.it

Department of Computer Science, University of Leicester
et52@mcs.le.ac.uk

1st CWI-DIPisa Workshop on Coordination, Coalgebras and Constraints
Pisa, January 21-23, 2008
The story

*SarA is a Software Architect. She develops software architectures with features such as style-conformance and dynamic reconfiguration. She meets AnDRea, a formal methodist, working on Architectural Design Rewriting.*
**SarA:** I need to design filter architectures. Some sort of pipelines that can be put in sequence or in parallel. Things like this diagram:

![Diagram of filter architectures](image)

**AnDRea:** I would call it a *graph*:

- components/connectors are hyperedges (boxes),
- ports/roles are tentacles (arrows),
- and attachments are nodes (circles).
**SarA:** I need a mechanism to build configurations in that *style* to avoid cyclic or broken flows.

**AnDRea:** A graph grammar provides you a style-consistent refinement process. For instance, this rule lets you refine a filter as a sequential composition of filters:
**SarA:** But I also need to compose designs.

**AnDRea:** You can use a right-to-left algebraic reading of the grammar. For instance, production rule

```
FILTER :::= FILTER FILTER FILTER
```

now becomes operation *(design production)*
**An operation?**

**AnDRea:** In your algebra...

- the sort is FILTER, an edge exposing two nodes,
- domain are *designs* - graphs with interfaces - of sort FILTER,
- an operation is like a design, where some edges are arguments,
- and substitution means *hyperedge replacement*.

*SarA prefers to skip such technical details.*
**SarA:** Filters can be put in parallel, with suitable components to dispatch and collect messages.

*AnDRea draws*...
SarA: I also need single filters and empty filters.
AnDRea draws...
AnDRea shows the whole picture...
**SarA:** Show me an example, please.

*AnDRea draws.*

\[ \text{seq}(\text{filter1}, \text{par}(\text{filter2}, \text{filter3})) \]
**SarA:** Show me an example, please.

*AnDRea draws.*

\[
\text{seq}(\text{filter1}, \text{par}(\text{filter2}, \text{filter3}))
\]
**SarA:** Show me an example, please.

*AnDRea draws.*

\[
\text{seq(}\text{filter1, par(}\text{filter2, filter3})\text{)}
\]
Interpreting a term

SarA: Show me an example, please.

AnDRea draws.

$$\text{seq}(\text{filter1}, \text{par}(\text{filter2}, \text{filter3}))$$
**SarA**: Show me an example, please.

*AnDRea draws.*

\[
\text{seq}(\text{filter1}, \text{par}(\text{filter2}, \text{filter3}))
\]
SarA: I also need a mechanism to manipulate configurations: to swap any sequence of filters or to serialise any parallel composition.

AnDRea: You can use graph transformation rules.

SarA: I don’t want to obtain things like unconnected filters. Is style preservation guaranteed by such rules?

AnDRea: You need to proof them style consistent with a theorem or using a semi-decidable procedure or...

SarA: Or?

AnDRea: You define rules at the level of terms, exploiting the structure they introduce.
**SarA:** Show me how to swap filters, please.

*AnDRea writes and draws...*

\[ \text{seq}(x_1, x_2) \rightarrow \text{seq}(x_2, x_1) \]
**SarA:** How can I reconfigure a sequence of three filters?

**AnDRea:** Rule \(\text{seq}(x_1, x_2) \rightarrow \text{seq}(x_2, x_1)\) has different instances:

- \(\text{seq}(\text{filter1}, \text{filter2}, \text{filter3})\)
  \(\rightarrow \text{seq}(\text{filter1}, \text{filter3}, \text{filter2})\)

*Hint:* \(\text{seq}\) is associative, \(\text{seq}(\text{filter1}, \text{seq}(\text{filter2}, \text{filter3})) = \text{seq}(\text{seq}(\text{filter1}, \text{filter2}), \text{filter3})\)
**Swapping filters**

**SarA:** How can I reconfigure a sequence of three filters?  
**AnDRea:** Rule \( \text{seq}(x_1,x_2) \rightarrow \text{seq}(x_2,x_1) \) has different instances:

- \( \text{seq}(\text{filter1,filter2,filter3}) \rightarrow \text{seq}(\text{filter1,filter3,filter2}) \)
- \( \text{seq}(\text{filter1,filter2,filter3}) \rightarrow \text{seq}(\text{filter2,filter3,filter1}) \)

**Hint:** \( \text{seq} \) is associative,  
\( \text{seq}(\text{filter1,seq}(\text{filter2,filter3})) = \text{seq}(\text{seq}(\text{filter1,filter2}),\text{filter3}) \)
SarA: How can I reconfigure a sequence of three filters?

AnDRea: Rule $\text{seq}(x_1,x_2) \rightarrow \text{seq}(x_2,x_1)$ has different instances:

- $\text{seq}(\text{filter1},\text{filter2},\text{filter3})$
  - $\rightarrow \text{seq}(\text{filter1},\text{filter3},\text{filter2})$
- $\text{seq}(\text{filter1},\text{filter2},\text{filter3})$
  - $\rightarrow \text{seq}(\text{filter2},\text{filter3},\text{filter1})$
- $\text{seq}(\text{filter1},\text{filter2},\text{filter3})$
  - $\rightarrow \text{seq}(\text{filter2},\text{filter1},\text{filter3})$
- $\text{seq}(\text{filter1},\text{filter2},\text{filter3})$
  - $\rightarrow \text{seq}(\text{filter3},\text{filter1},\text{filter2})$

Hint: $\text{seq}$ is associative, $\text{seq}(\text{filter1},\text{seq}(\text{filter2},\text{filter3})) = \text{seq}(\text{seq}(\text{filter1},\text{filter2}),\text{filter3})$
**SarA:** How can I reconfigure a sequence of three filters?

**AnDRea:** Rule \( \text{seq}(x_1, x_2) \rightarrow \text{seq}(x_2, x_1) \) has different instances:

- \( \text{seq}(\text{filter1, filter2, filter3}) \rightarrow \text{seq}(\text{filter1, filter3, filter2}) \)
- \( \text{seq}(\text{filter1, filter2, filter3}) \rightarrow \text{seq}(\text{filter2, filter3, filter1}) \)
- \( \text{seq}(\text{filter1, filter2, filter3}) \rightarrow \text{seq}(\text{filter2, filter1, filter3}) \)
- \( \text{seq}(\text{filter1, filter2, filter3}) \rightarrow \text{seq}(\text{filter3, filter1, filter2}) \)

*Hint:* \( \text{seq} \) is associative, \( \text{seq}(\text{filter1, seq(filter2, filter3)}) = \text{seq}(\text{seq(filter1, filter2), filter3}) \)
**SarA:** And how are parallel filters serialised?

*AnDRea writes and draws...*

\[
\text{par}(x_1,x_2) \rightarrow \text{seq}(x_1,x_2)
\]
**SarA**: I need serialisation to be propagated...

**AnDRea**: Conditional labelled rules do the job

**AnDRea**: Other rules are needed to propagate and stop the propagation.
**SarA:** Show me an example, please.

*AnDRea draws...*
SarA: What is ADR?
- Algebra of graphs with interfaces (designs).
- Conditional, labelled rewrite rules on design terms.

SarA: What can I do with ADR?
- Algebra = architectural style, metamodel, graphical encoding.
- Rewrite rules = reconfiguration, dynamic binding, ordinary execution.

SarA: What is the status of ADR?
- Other domains of interpretation: constraints?
- Specification and Verification: e.g. exploiting the structure.
- Prototypical implementation in Maude.

SarA: Where can I get more information on ADR?
SarA: What about Coordination? Coalgebras? Constraints?
AnDRea: Let’s talk about it later. Let’s say... at dinner?