Architectural Design Rewriting

R. Bruni[1], A. Lluch Lafuente[1], U. Montanari[1], E. Tuosto[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

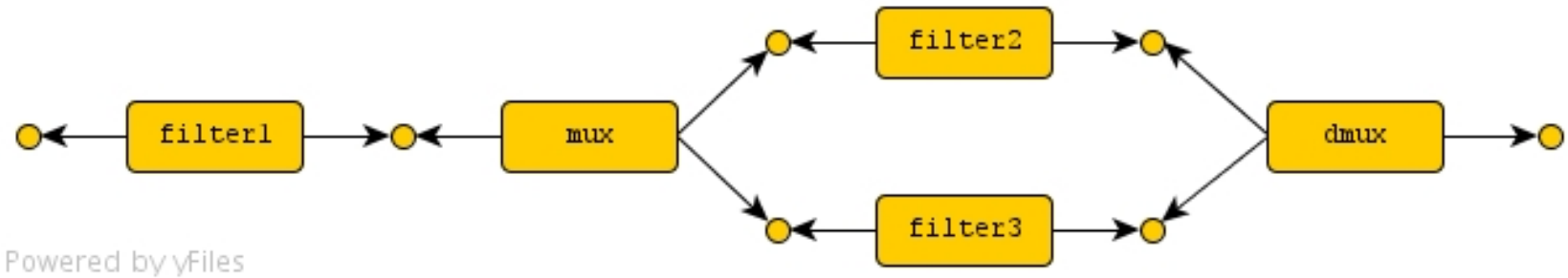
1st CWI-DIPisa Workshop on
Coordination, Coalgebras and Constraints
Pisa, January 21-23, 2008

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.

Software Architectures, Graphs

SarA: I need to design filter architectures. Some sort of pipelines that can be put in sequence or in parallel. Things like this diagram:



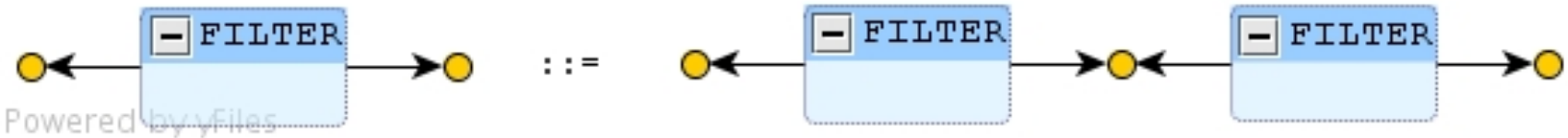
AnDRea: I would call it a *graph*:

- components/connectors are hyperedges (boxes),
- ports/roles are tentacles (arrows),
- and attachments are nodes (circles).

Architectural Styles, Graph Grammars

SarA: I need a mechanism to build configurations in that *style* to avoid cyclic or broken flows.

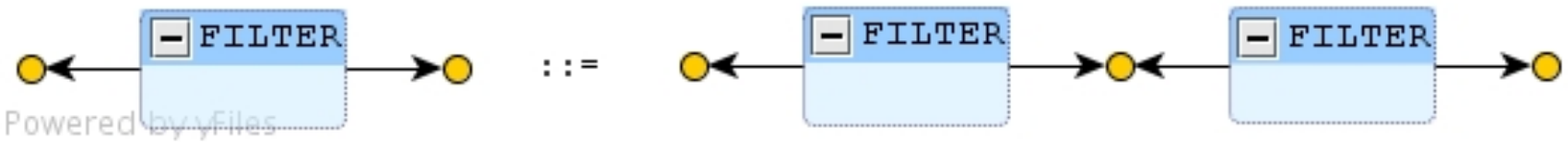
AnDRea: A graph grammar provides you a style-consistent refinement process. For instance, this rule lets you refine a filter as a sequential composition of filters:



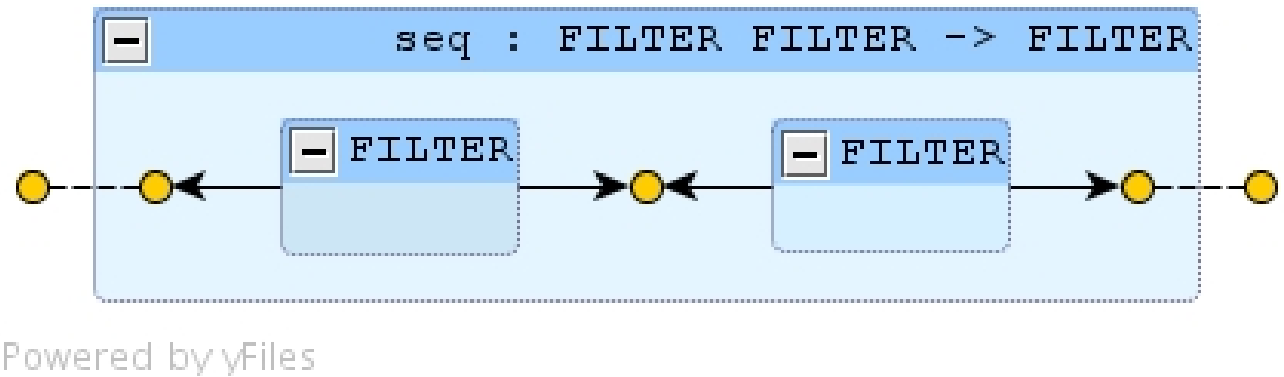
Architectural Styles, From Grammars to Algebras

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



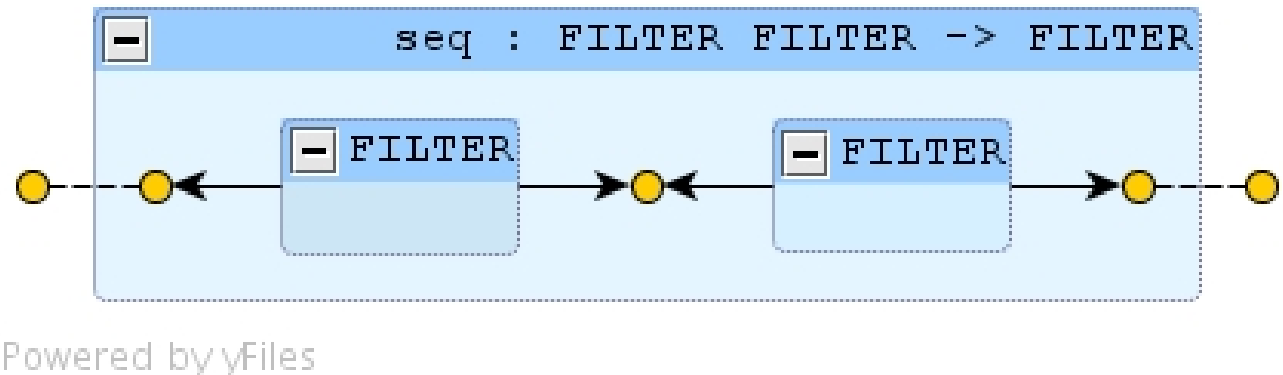
now becomes operation (*design production*)



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*.

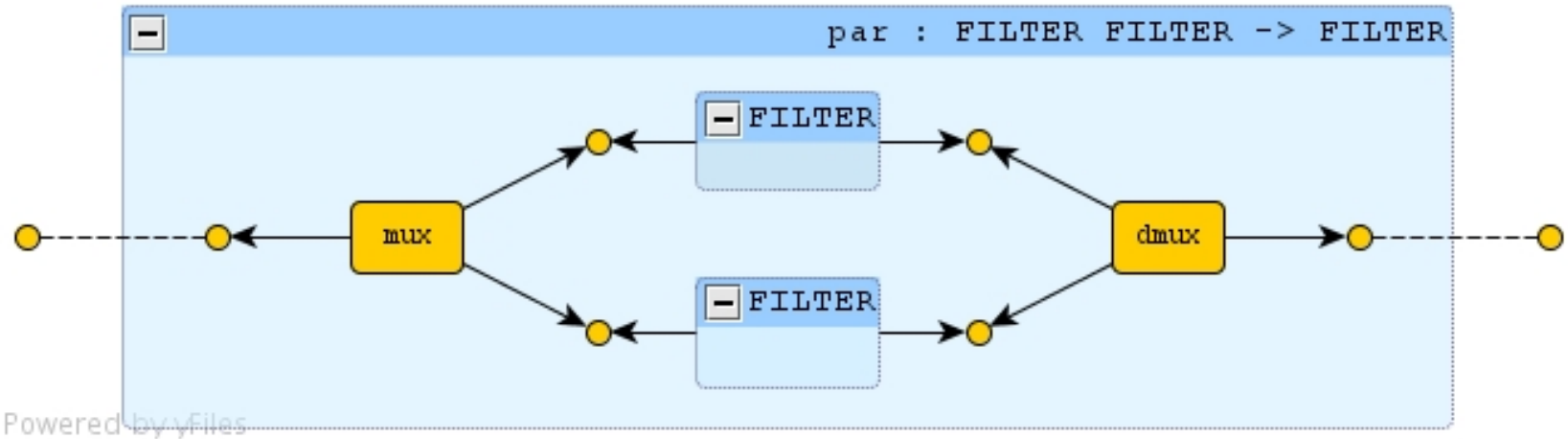


SarA prefers to skip such technical details.

More productions: parallel filters

SarA: Filters can be put in parallel, with suitable components to dispatch and collect messages.

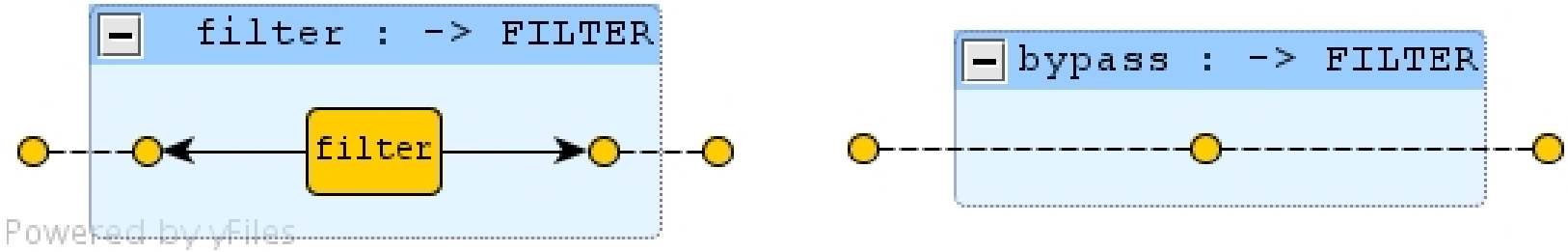
AnDRea draws...



More productions: empty and single filters

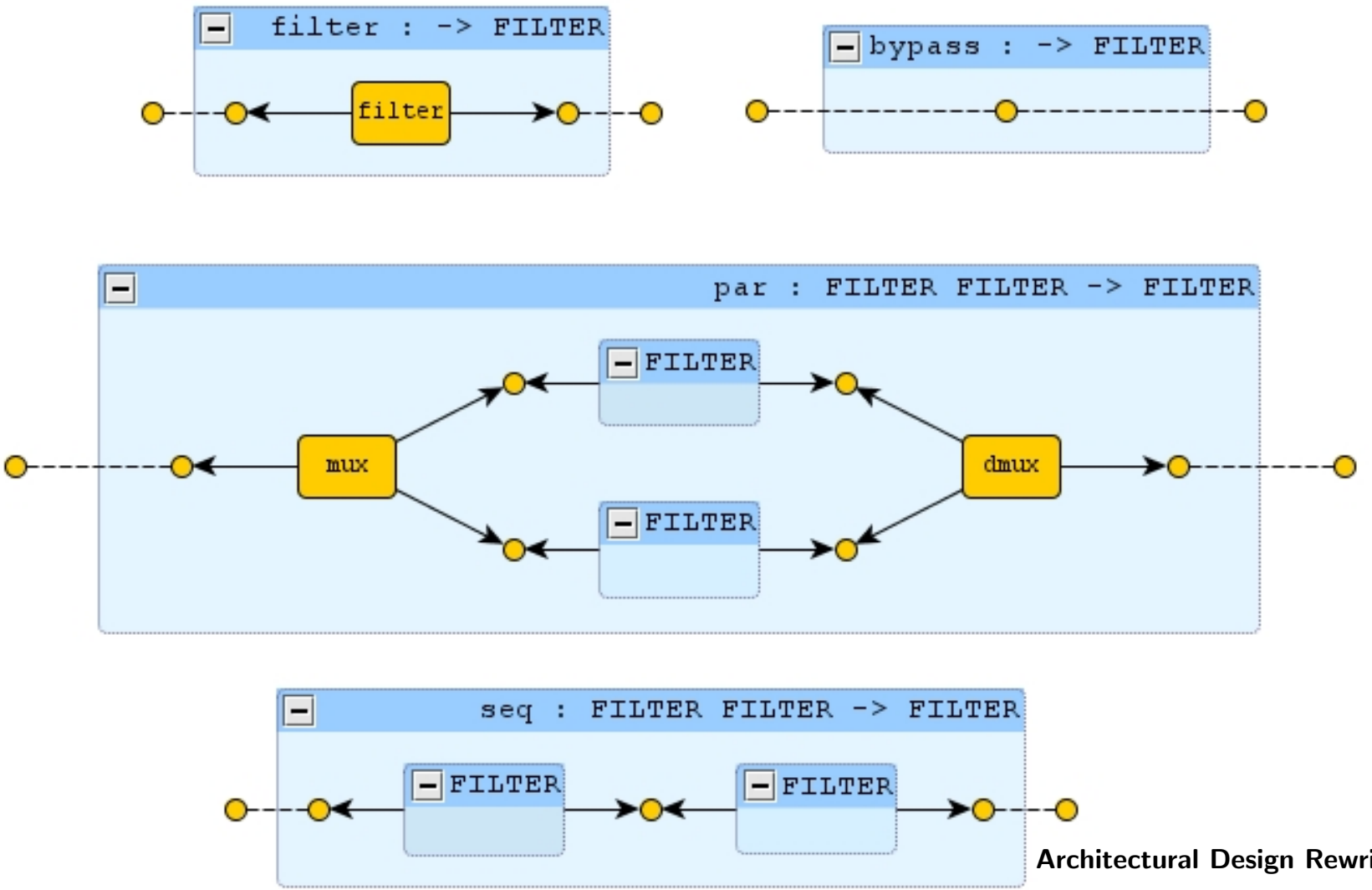
SarA: I also need single filters and empty filters.

AnDRea draws...



All productions

AnDRea shows the whole picture...

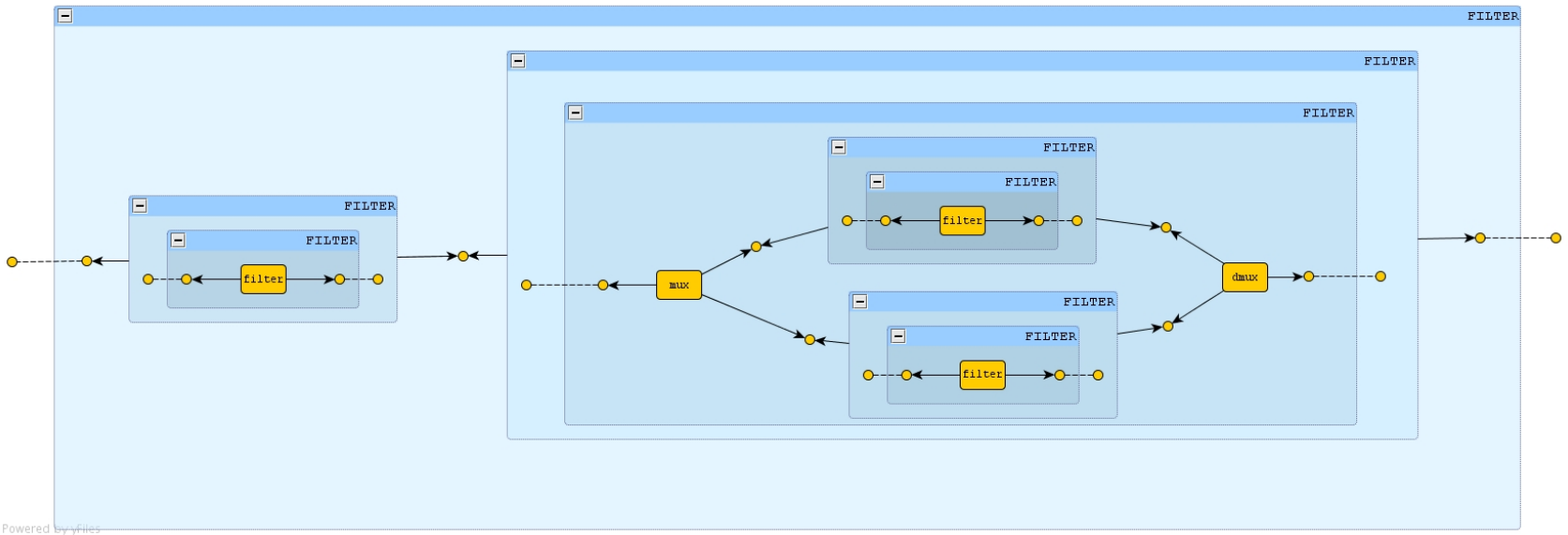


Interpreting a term

SarA: Show me an example, please.

AnDRea draws.

```
seq(filter1,par(filter2,filter3))
```

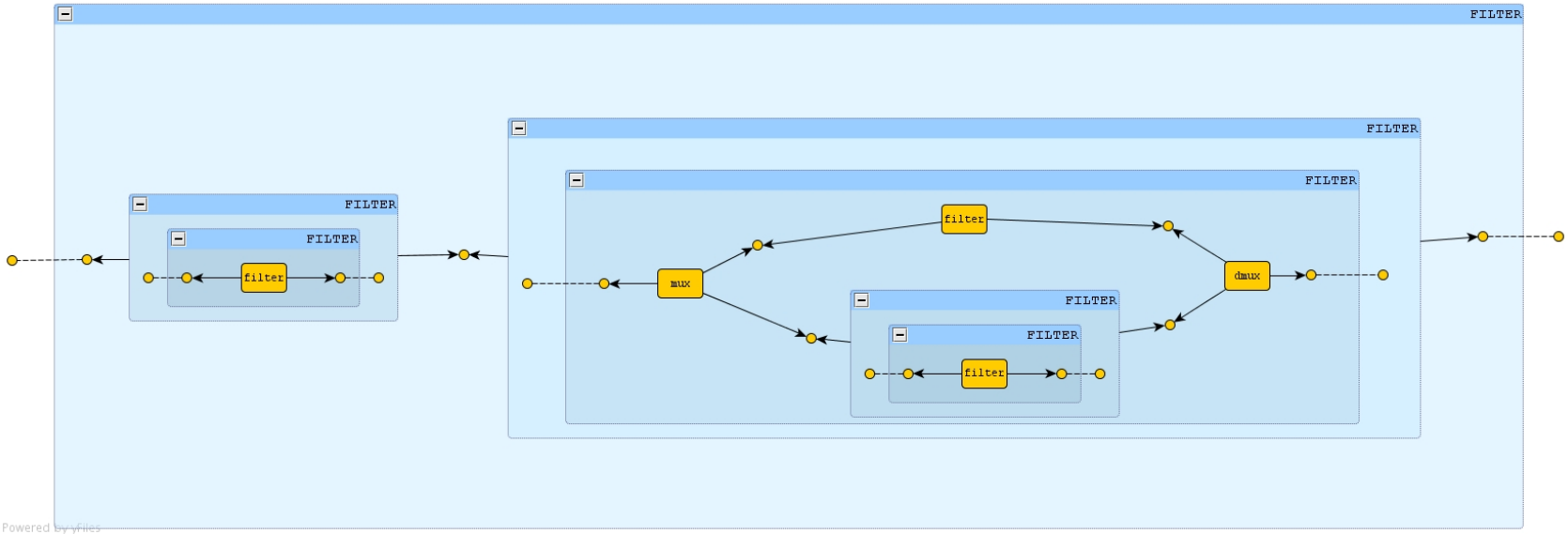


Interpreting a term

SarA: Show me an example, please.

AnDRea draws.

```
seq(filter1,par(filter2,filter3))
```

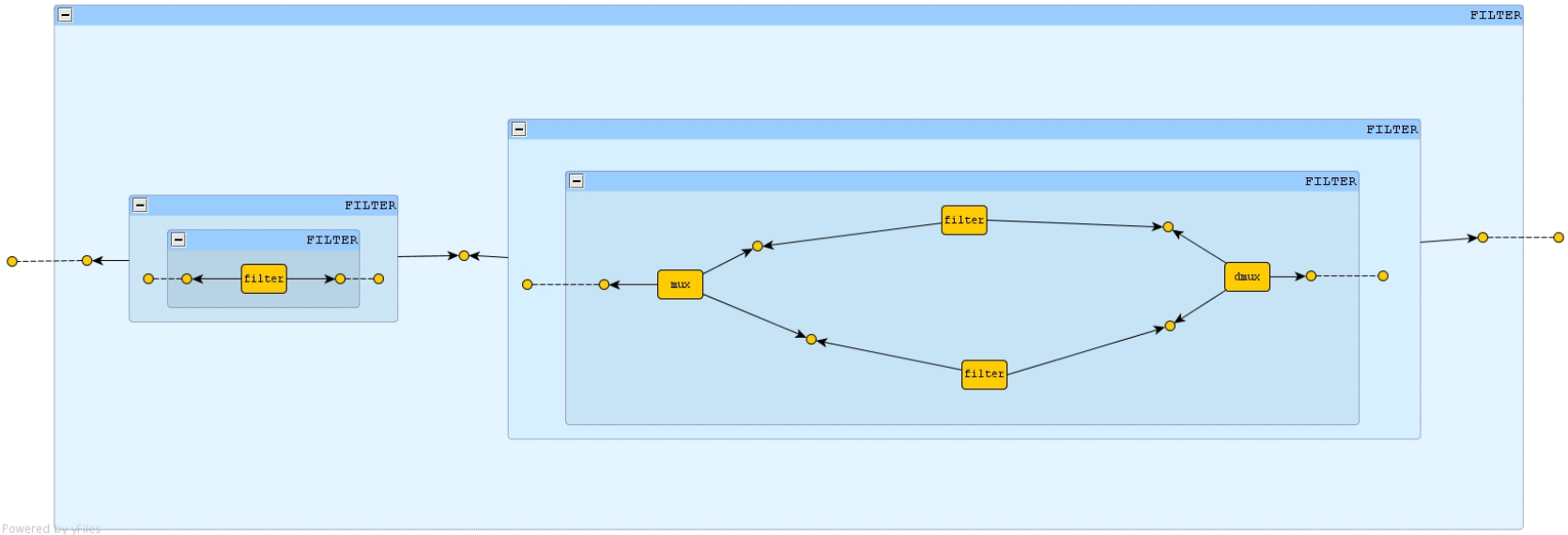


Interpreting a term

SarA: Show me an example, please.

AnDRea draws.

```
seq(filter1,par(filter2,filter3))
```

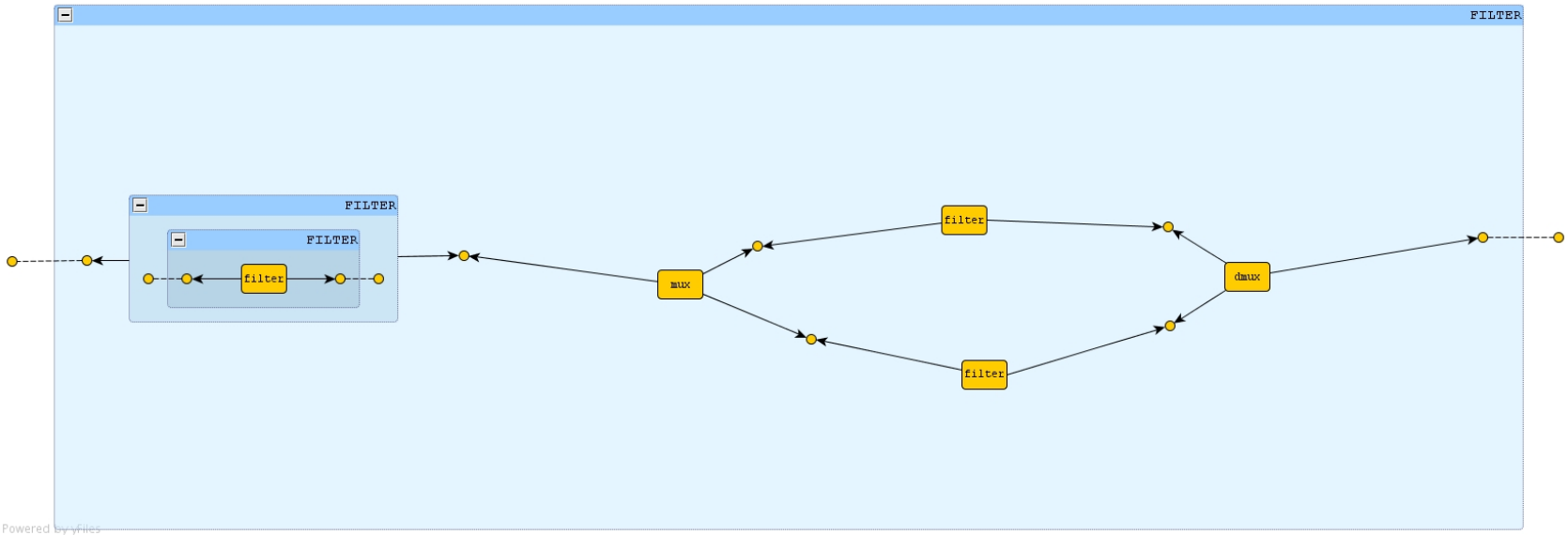


Interpreting a term

SarA: Show me an example, please.

AnDRea draws.

```
seq(filter1,par(filter2,filter3))
```

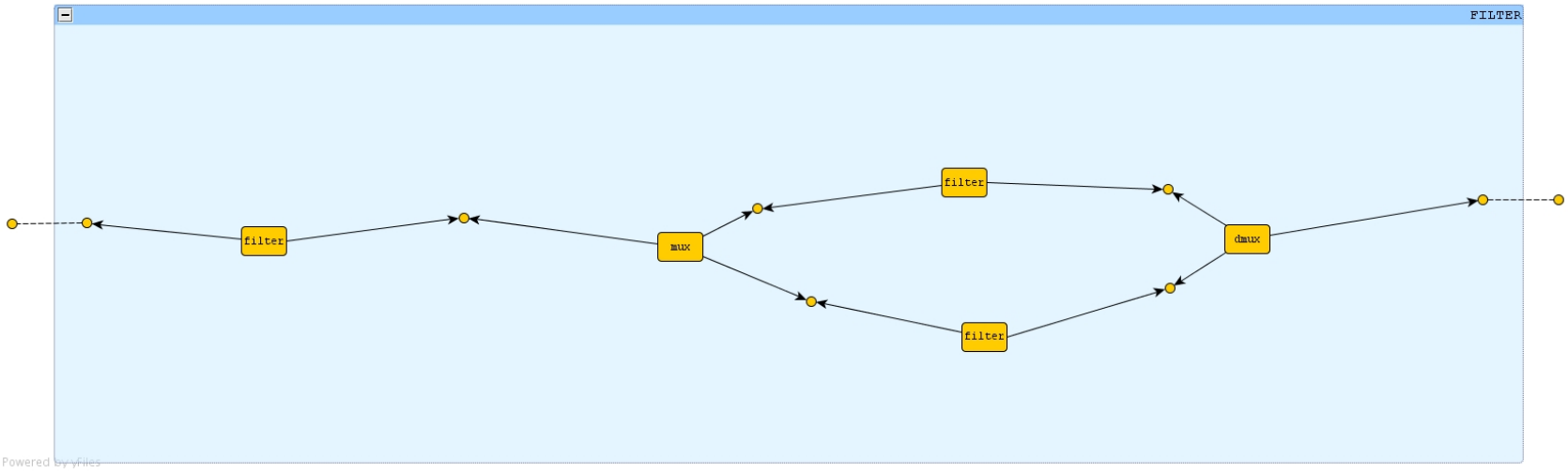


Interpreting a term

SarA: Show me an example, please.

AnDRea draws.

```
seq(filter1,par(filter2,filter3))
```



Reconfigurations

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.

Swapping filters

SarA: Show me how to swap filters, please.

AnDRea writes and draws...

```
seq(x1,x2) -> seq(x2,x1)
```

Swapping filters

SarA: How can I reconfigure a sequence of three filters?

AnDRea: Rule `seq(x1,x2) -> seq(x2,x1)` has different instances:

- ```
  seq(filter1,filter2,filter3)
  -> seq(filter1,filter3,filter2)
  ```

Hint: `seq` is associative, `seq(filter1,seq(filter2,filter3))` = `seq(seq(filter1,filter2),filter3)`
  ```

Swapping filters

SarA: How can I reconfigure a sequence of three filters?

AnDRea: Rule `seq(x1,x2) -> seq(x2,x1)` has different instances:

- `seq(filter1,filter2,filter3)`
  `-> seq(filter1,filter3,filter2)`
- `seq(filter1,filter2,filter3)`
  `-> seq(filter2,filter3,filter1)`

Hint: `seq` is associative, `seq(filter1,seq(filter2,filter3))` = `seq(seq(filter1,filter2),filter3)`

Swapping filters

SarA: How can I reconfigure a sequence of three filters?

AnDRea: Rule `seq(x1,x2) -> seq(x2,x1)` has different instances:

- `seq(filter1,filter2,filter3)`
  `-> seq(filter1,filter3,filter2)`
- `seq(filter1,filter2,filter3)`
  `-> seq(filter2,filter3,filter1)`
- `seq(filter1,filter2,filter3)`
  `-> seq(filter2,filter1,filter3)`

Hint: `seq` is associative, `seq(filter1,seq(filter2,filter3))` = `seq(seq(filter1,filter2),filter3)`

Swapping filters

SarA: How can I reconfigure a sequence of three filters?

AnDRea: Rule `seq(x1,x2) -> seq(x2,x1)` has different instances:

- `seq(filter1,filter2,filter3)`
  `-> seq(filter1,filter3,filter2)`
- `seq(filter1,filter2,filter3)`
  `-> seq(filter2,filter3,filter1)`
- `seq(filter1,filter2,filter3)`
  `-> seq(filter2,filter1,filter3)`
- `seq(filter1,filter2,filter3)`
  `-> seq(filter3,filter1,filter2)`

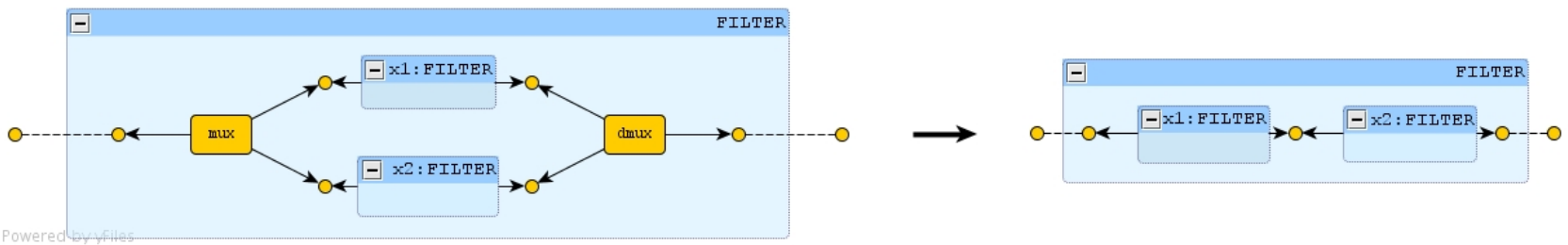
Hint: `seq` is associative, `seq(filter1,seq(filter2,filter3))` = `seq(seq(filter1,filter2),filter3)`

Serialisation

SarA: And how are parallel filters serialised?

AnDRea writes and draws...

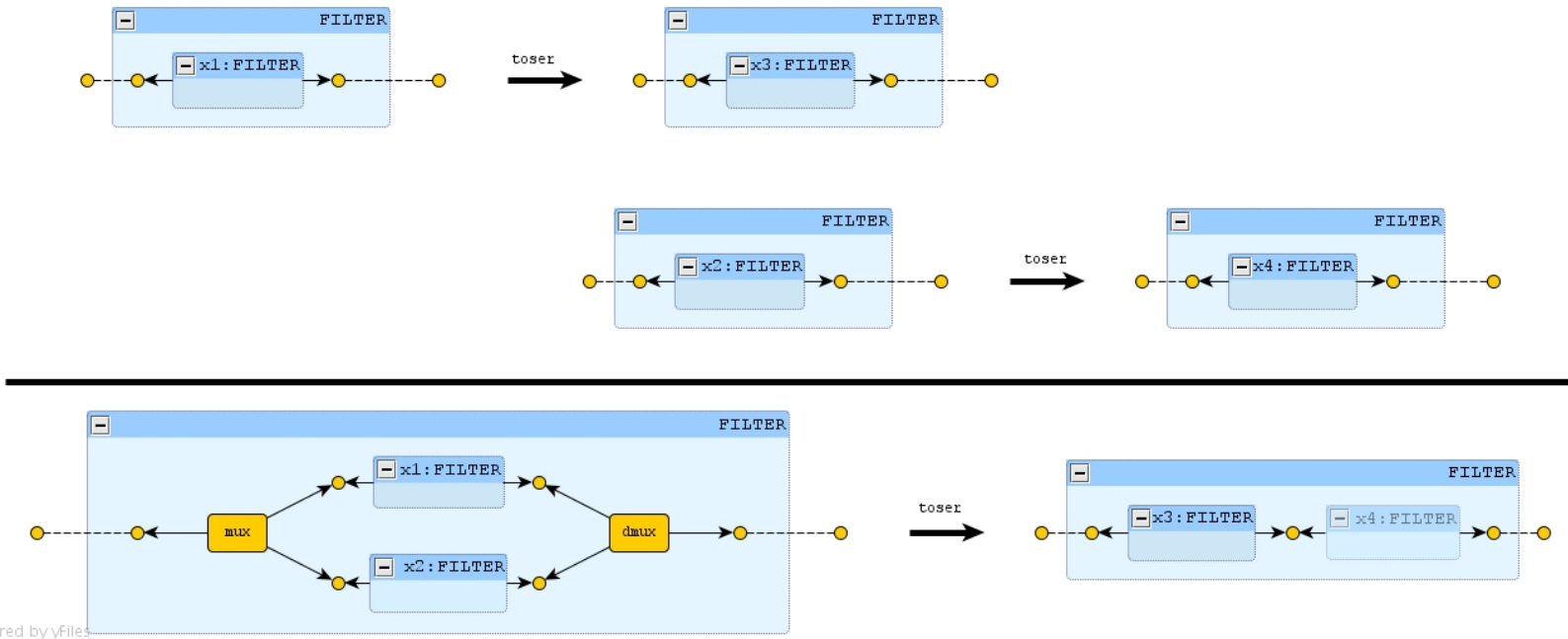
```
par(x1,x2) -> seq(x1,x2)
```



Propagated Serialisation

SarA: I need serialisation to be propagated...

AnDRea: Conditional labelled rules do the job

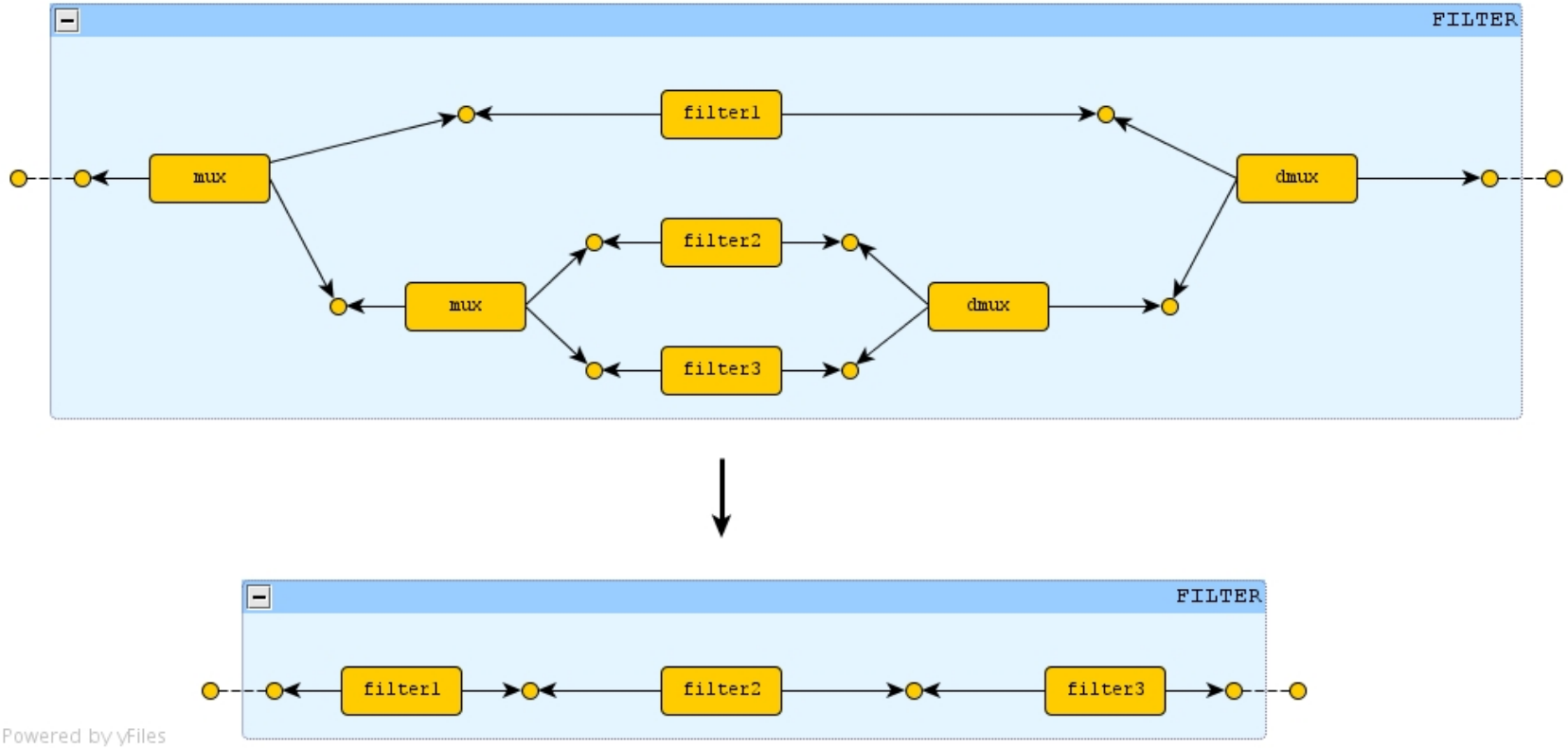


AnDRea: Other rules are needed to propagate and stop the propagation.

Propagated Serialisation

SarA: Show me an example, please.

AnDRea draws...



Architectural Design Rewriting

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?

- `www.albertolluch.com/adr.html`.

Epilogue

SarA: What about Coordination? Coalgebras? Constraints?
AnDRea: Let's talk about it later. Let's say... at dinner?