

On Dynamic Reconfigurations in Reo and the Eclipse Coordination Tools

Christian Krause

Centrum Wiskunde & Informatica (CWI), Amsterdam

CIC Workshop - 7 May 2009

Outline

Dynamic Reconfigurations in Reo

Eclipse Coordination Tools

Reconfiguration

- Formalised using alg. graph transformation: [KMLA09]
- Distributed version described in [KAV08]

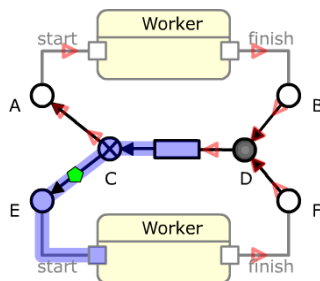


Figure: an example network

Reconfiguration Rules

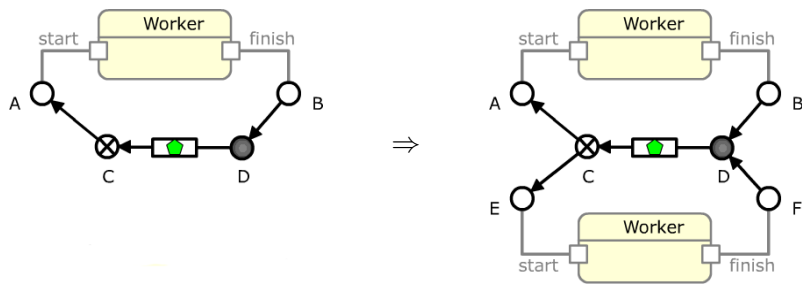


Figure: example reconfiguration rule: *AddWorker*

- *DelWorker* is defined as the inverse rule

Reconfiguration Rules

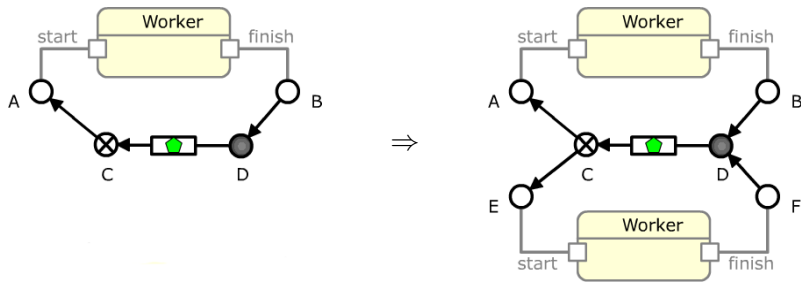


Figure: example reconfiguration rule: *AddWorker*

- *DelWorker* is defined as the inverse rule

Reconfiguration Rules

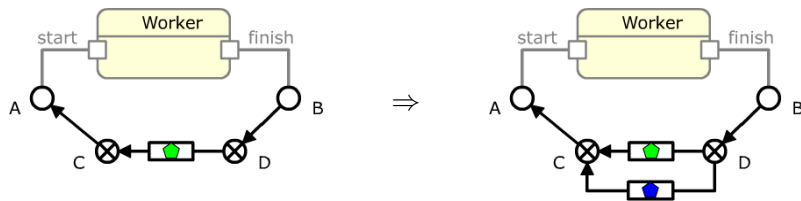


Figure: example reconfiguration rule: *AddResource*

- *DelResource* is defined as the inverse rule

Reconfiguration Rules

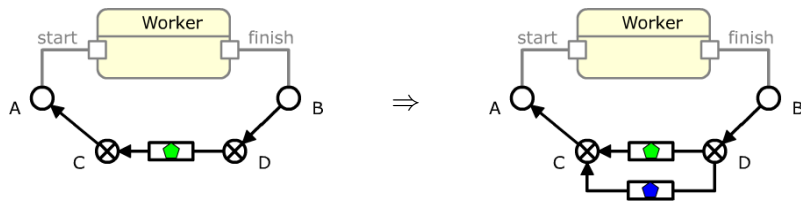


Figure: example reconfiguration rule: *AddResource*

- *DelResource* is defined as the inverse rule

Export to AGG

- <http://tfs.cs.tu-berlin.de/agg>

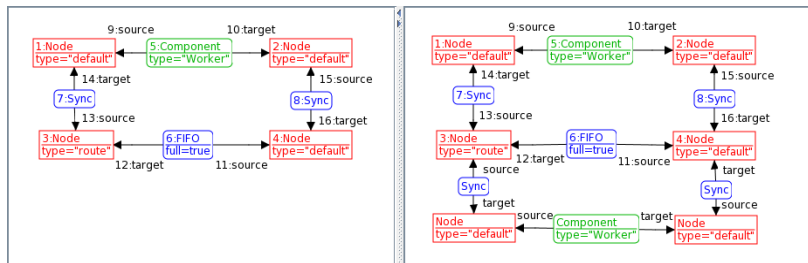
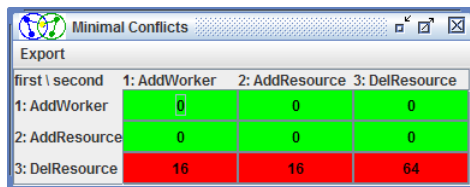


Figure: exported rule *AddWorker*

Analysis in AGG

- Critical-Pair analysis:



first \ second	1: AddWorker	2: AddResource	3: DelResource
1: AddWorker	0	0	0
2: AddResource	0	0	0
3: DelResource	16	16	64

Figure: Results of a critical pair analysis in AGG

Invariants

Verifying structural and behavioral invariants:

1. *There is at least one worker and one FIFO1 buffer for a resource.*
2. *The number of empty FIFO1s equals to the number of active workers.*

⇒ 2. can be checked only if the primitive's states are considered as well

Invariants

Verifying structural and behavioral invariants:

1. *There is at least one worker and one FIFO1 buffer for a resource.*
2. *The number of empty FIFO1s equals to the number of active workers.*

⇒ 2. can be checked only if the primitive's states are considered as well

Refined Reconfiguration Rule

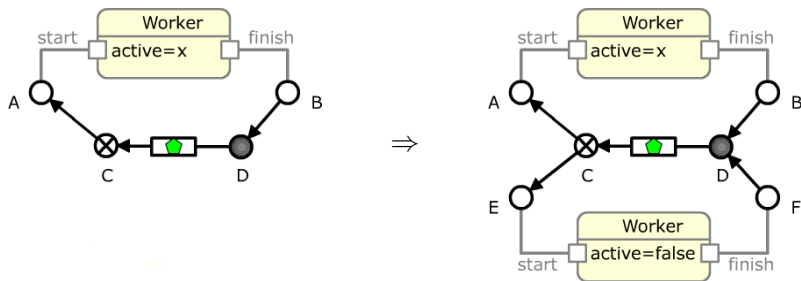


Figure: refined reconfiguration rule: *AddResource*

Execution Rule

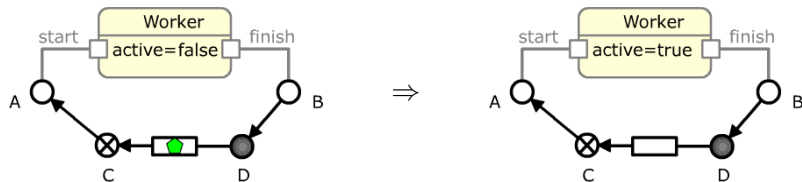


Figure: execution rule: *StartWorker*

- *StopWorker* is defined as the inverse rule

Execution Rule

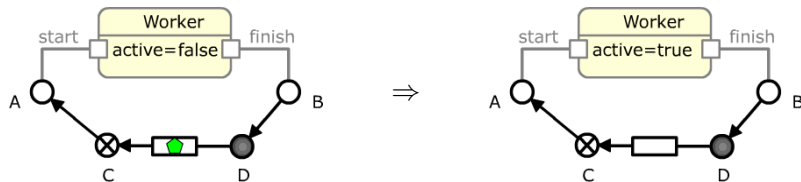


Figure: execution rule: *StartWorker*

- *StopWorker* is defined as the inverse rule

State Space Analysis in GROOVE

- <http://groove.cs.utwente.nl/>

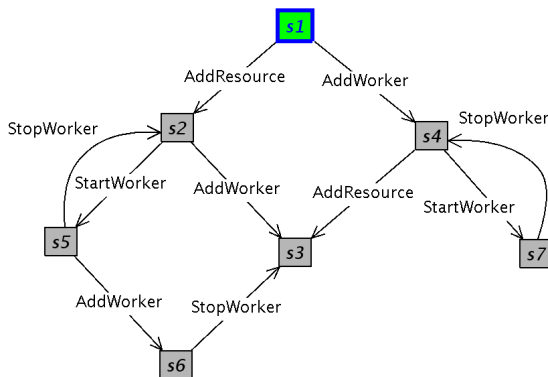


Figure: A manual state space exploration in GROOVE

Model checking in GROOVE

- $AG(\text{StartWorker} \rightarrow EX(\text{StopWorker}))$: After starting a worker it can always be stopped again in the next step.
- $AG(\text{AddResource})$: It is always possible to add a resource.
- $AG(\text{StopWorker} \rightarrow EX(\text{AddResource}))$: After stopping a worker it is always possible to add a resource in the next step.

Problems

- No compositional graph transformation semantics for Reo.
- Still there is a need to analyse the relation of execution and rule-based reconfigurations.
- Port / constraint automata are a good model for analysis and centralised execution, but:
 - analysis of dynamic reconfigurations requires a combined structural and behavioral model
 - dynamic reconfigurations in constraint automata approach are inefficient and require a *state recovery*

Future Work

- Categorical model of port automata
- Define a (contravariant) functor from Reo graphs to port automata
- Compositionality: the functor turns pushouts of graphs into pullbacks of automata
- (Encode Petri nets in this model)

Graph Rewriting

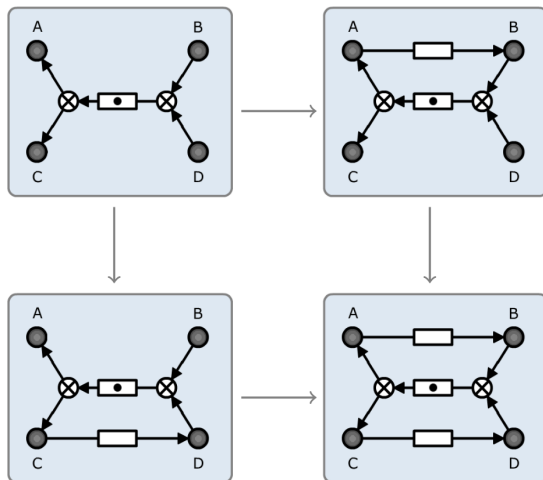


Figure: A pushout of Reo graphs

Automata Rewriting

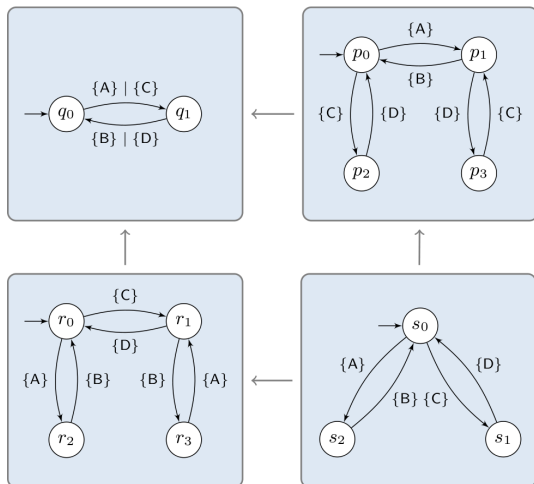




Figure: Corresponding pullback of port automata

References

-  C. Krause, Z. Maraikar, A. Lazovik, F. Arbab.
Modeling Dynamic Reconfigurations in Reo using High-Level Replacement Systems.
To appear in Science of Computer Programming.
-  C. Koehler, F. Arbab, E. de Vink.
Reconfiguring Distributed Reo Connectors.
To appear in proceedings of WADT 2008 (LNCS).

Eclipse Coordination Tools

- Detailed info at: <http://reo.project.cwi.nl>
- Version 3.2.0 released on 23 April

- **Reo Core Tools:**
 - connector model
 - graphical editor
 - animation tool

- **Vereofy Model Checker:**
 - <http://www.vereofy.de>

Eclipse Coordination Tools

- **Reo Automata Tools (EA):**
 - extensible automata model (automata extensions/types and product definitions)
 - graphical editor

- **Reo to Automata Conversion (Reo2EA):**
 - extensible converter
 - currently supported: CA and QIA

Eclipse Coordination Tools

- **Reo Runtime and Code Generation:**
 - Java code generator
 - experimental C code generator (not released)
 - CA interpreter

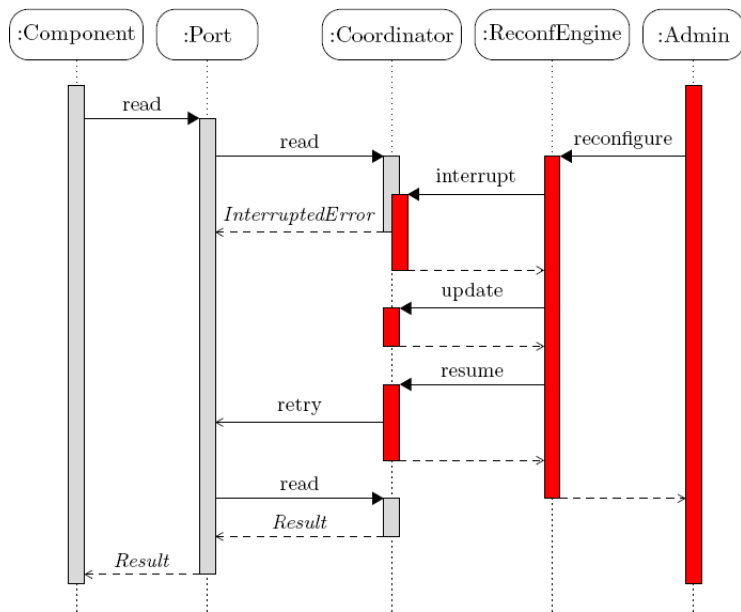
- **Reo Distributed Engine:**
 - distributed engine impl. in Scala
 - direct deployment from the Reo editor possible

Eclipse Coordination Tools

- **Reo Reconfiguration Support:**
 - definition and in-place application of rules
 - export to AGG

- **ReoLive Coordination Web Service:**
 - <http://reoproject.cwi.nl/live>
 - centralised coordinator, distributed components
 - support for hot rewiring and dynamic reconfigurations

Executing Reconfigurations



References



F. Arbab et al.

Modeling, Testing and Executing Reo Connectors with the Eclipse Coordination Tools.

Proceedings of FACS 2008.