Behaviour agreement

José Proença

Centrum voor Wiskunde en Informatica

CIC, 2007





Motivation

Distributed Reo Behaviour agreement Conclusions

Outline

Motivation

- Motivation
- Distributed Reo Model
- Behaviour agreement
- 4 Conclusions



2/15



Distributed Reo Behaviour agreement Conclusions oo o

Outline

Motivation

- Motivation
- Distributed Reo Mode
- Behaviour agreement
- Conclusions



3 / 15



Distributed Reo Behaviour agreement

Motivation

Motivation



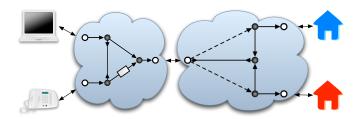
Coordination

- How to implement it?
- Where to run it?

Distributed Coordination







Coordination:

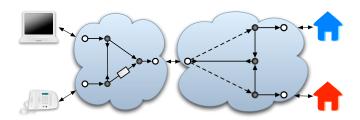
- How to implement it?
- Where to run it?

Distributed Coordination





J. Proença (CWI)



Coordination:

- How to implement it?
- Where to run it?

Distributed Coordination





Designer

Deployment Resolver

Local Optimization

Instantiator





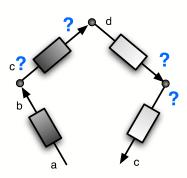
Designer

Use of tools, such as a GUI

Deployment Resolver

Local Optimization

Instantiator







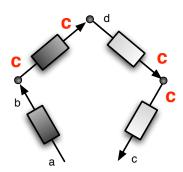
Designer

Deployment Resolver

Unspecified locations are resolved. Constraints and policies need to be considered.

Local Optimization

Instantiator







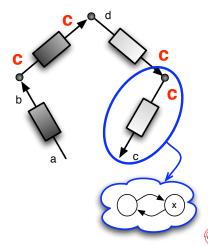


Deployment Resolver

Local Optimization

Plugins: CA CC CSP

Instantiator



Designer

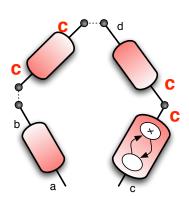
Deployment Resolver

Local Optimization

Instantiator

Creation of primitives

Kernel





5/15



J. Proença (CWI) Towards Distributed Reo

Designer

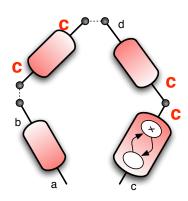
Deployment Resolver

Local Optimization

Instantiator

Kernel

Execution of the engine











Behaviour agreement

Distributed Reo

Outline

- Distributed Reo Model



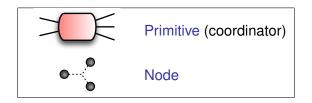
6/15



Behaviour agreement Conclusions

Distributed Reo

Distributed: deals with partial knowledge.





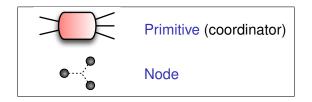


Towards Distributed Reo J. Proenca (CWI) CIC, 2007 7/15

Behaviour agreement Conclusions

Distributed Reo

Distributed: deals with partial knowledge.



Implemention: Scala language

Integrates features of object-oriented and functional languages; Fully interoperable with Java;

Actor model for communication.





J. Proenca (CWI)

Distributed Reo

Primitives and nodes



Distributed Reo

Each port has a *location*. Must react to some messages:

- Request Behabiour
- Reply Behaviour
- Refuse (reason)
- Give Behaviour & Request/Give Data
- Reply Data



- has no state:
- can be distributed:
- propagates synchronous





Conclusions

Distributed Reo

Primitives and nodes



Each port has a *location*. Must react to some messages:

- Request Behabiour
- Reply Behaviour
- Refuse (reason)
- Give Behaviour & Request/Give Data
- Reply Data



Can be seen as a particular case of a primitive that:

- has no state;
- can be distributed;
- propagates synchronous constraints.







Behaviour What is it?

What each primitive can do

Which end points can flow data, and relation between data flowing in the end points.

Join of behaviours

Given the behaviour of two primitives, the behaviour of the composition of both can also be obtained.

Example: Connector Colouring

- Colouring tables provide the behaviour of each primitive;
- Join of colouring tables is defined





Behaviour agreement Conclusions

What is it?

Behaviour

What each primitive can do

Which end points can flow data, and relation between data flowing in the end points.

Join of behaviours

Given the behaviour of two primitives, the behaviour of the composition of both can also be obtained.

Example: Connector Colouring

- Colouring tables provide the behaviour of each primitive;
- Join of colouring tables is defined.





Distributed Reo Behaviour agreement Conclusions

Outline

Motivation

- Motivation
- Distributed Reo Mode
- Behaviour agreement
- 4 Conclusions

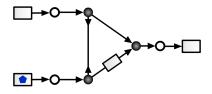


10 / 15



Locations are not relevant: Assume partial knowledge (know only neighbours); Two phase algorithm: Negotiation and Communication.

Behaviour agreement







Commit to a behaviour

Locations are not relevant;

Assume partial knowledge (know only neighbours);

Two phase algorithm: Negotiation and Communication.



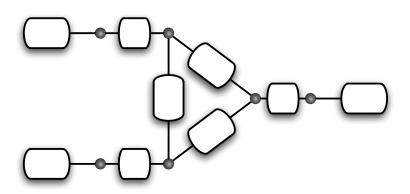


J. Proença (CWI) Towards Distributed Reo

 Motivation
 Distributed Reo
 Behaviour agreement
 Conclusions

 ○○
 ○○
 ○●○
 ○

Commit to a behaviour Basic case



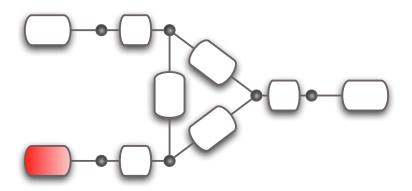




otivation Distributed Reo Behaviour agreement Conclusions

○ ○ ○ ○ ○ ○ ○ ○ ○

Commit to a behaviour Basic case

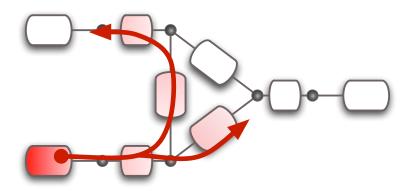






Motivation Distributed Reo Behaviour agreement Conclusions
oo oo oo oo oo oo

Commit to a behaviour Basic case

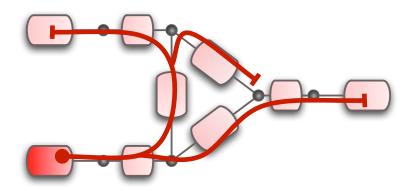






Motivation Distributed Reo Behaviour agreement Conclusions
oo oo oo oo oo oo

Commit to a behaviour Basic case





12 / 15



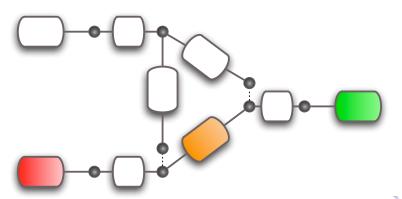
Motivation Distributed Reo Behaviour agreement Conclusions

○○ ○○ ○○ ○○ ○● ○ ○

Commit to a behaviour

Multiple starting points







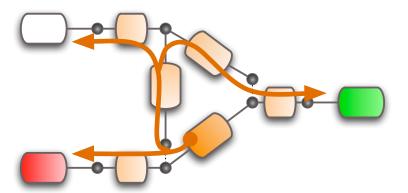
 Motivation
 Distributed Reo
 Behaviour agreement
 Conclusions

 ○○
 ○○
 ○○
 ○

Commit to a behaviour

Multiple starting points







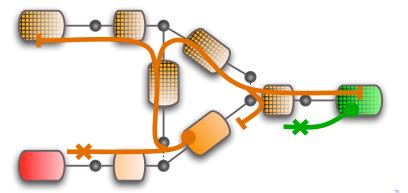
Motivation Distributed Reo Behaviour agreement Conclusions

○○ ○○ ○○ ○○ ○● ○

Commit to a behaviour

Multiple starting points







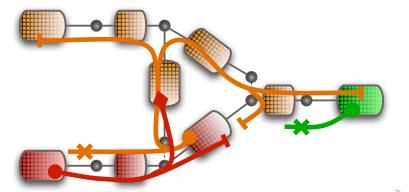
Motivation Distributed Reo Behaviour agreement Conclusions

○○ ○ ○ ○ ○ ○ ○ ○ ○ ○ ○

Commit to a behaviour

Multiple starting points







2007 13 / 15

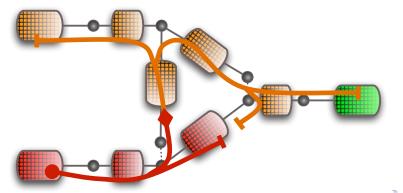
 Motivation
 Distributed Reo
 Behaviour agreement
 Conclusions

 ○○
 ○○
 ○○
 ○

Commit to a behaviour

Multiple starting points







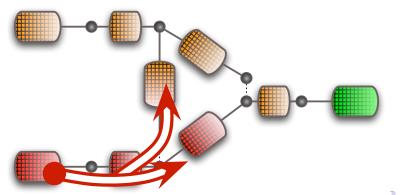
Motivation Distributed Reo Behaviour agreement Conclusions

○○ ○○ ○○ ○○ ○○ ○○ ○ ○

Commit to a behaviour

Multiple starting points







Distributed Reo Behaviour agreement Conclusions

Outline

Motivation

- Motivation
- Distributed Reo Model
- Behaviour agreement
- 4 Conclusions



14 / 15



- Common architecture to include design and implementation;
- Implementation platform, where each (distributed) element knows only about its own neighbours;
- Resolve synchrony constraints (imposed by Reo) using asynchronous messages;
- The kernel supports messages for other purposes:
 - Fail/Abort;
 - Suspend to allow reconfiguration.
 - . . .
- How to determine the rank of the inititiators?
- A primitive(s) can be obtained from other coordination models other than Reo (e.g., Orc);
- Allow unification of more coordination models:



J. Proenca (CWI) Towards Distributed Reo CIC, 2007 15 / 15

- Common architecture to include design and implementation;
- Implementation platform, where each (distributed) element knows only about its own neighbours;
- Resolve synchrony constraints (imposed by Reo) using asynchronous messages;
- The kernel supports messages for other purposes:
 - Fail/Abort:
 - Suspend to allow reconfiguration.
 - ...
- How to determine the rank of the *inititiators*?
- A primitive(s) can be obtained from other coordination models
- Allow unification of more coordination models:



15 / 15

Towards Distributed Reo

- Common architecture to include design and implementation;
- Implementation platform, where each (distributed) element knows only about its own neighbours;
- Resolve synchrony constraints (imposed by Reo) using asynchronous messages;
- The kernel supports messages for other purposes:
 - Fail/Abort;
 - Suspend to allow reconfiguration.
 - ...
- How to determine the rank of the inititiators?
- A primitive(s) can be obtained from other coordination models other than Reo (e.g., Orc);
- Allow unification of more coordination models:



J. Proenca (CWI) Towards Distributed Reo CIC, 2007 15 / 15

- Common architecture to include design and implementation;
- Implementation platform, where each (distributed) element knows only about its own neighbours;
- Resolve synchrony constraints (imposed by Reo) using asynchronous messages;
- The kernel supports messages for other purposes:
 - Fail/Abort;
 - Suspend to allow reconfiguration.
 - ...
- How to determine the rank of the inititiators?
- A primitive(s) can be obtained from other coordination models other than Reo (e.g., Orc);
- Allow unification of more coordination models:



J. Proenca (CWI) Towards Distributed Reo CIC, 2007 15 / 15