# Reo Connector semantics using Connector Colouring

David de Oliveira Costa
CWI, Amsterdam, Netherlands
costa@cwi.nl
http://www.cwi.nl/~costa

October 12, 2006

# Overview

- **Motivation**
- **Reo Connectors**
- **Connector Colouring**
  - Terminology
  - 2-colouring model
    - properties
  - 3-colouring model
    - properties
- **Ongoing work**
  - Distributed implementation
  - Animation of Reo connectors
  - Connector Calculus
- **Conclusion**

# Motivation

- **Existing Semantic Models for Reo**
  - Constraint Automata (most robust and mature)
  - Abstract Behaviour Types and Coinductive Calculus

# Motivation

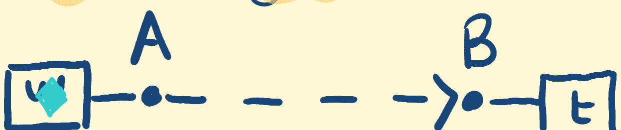
- **Pros:**
  - Clear and simple model of the observational behaviour
  - Compositional
  - Provide formal support to reason about the model:
    - Equivalence (Bisimulation and Language equivalence)
    - Refinement (Simulation and Language inclusion)
    - Model Checking
    - Etc...

# Motivation

- **Existing Semantic Models for Reo**
  - Constraint Automata
  - Coinductive Calculus
  
  and Abstract Behaviour Types

Don't model accurately Reo connectors that have *context sensitive behaviour*

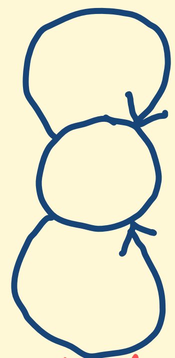
LossySync




It forces the data through the channel if we have I/O operations on both ends A and B

It loses the data input in A in case no read operation is present in B

$\{A,B\}, d_A = d_B$

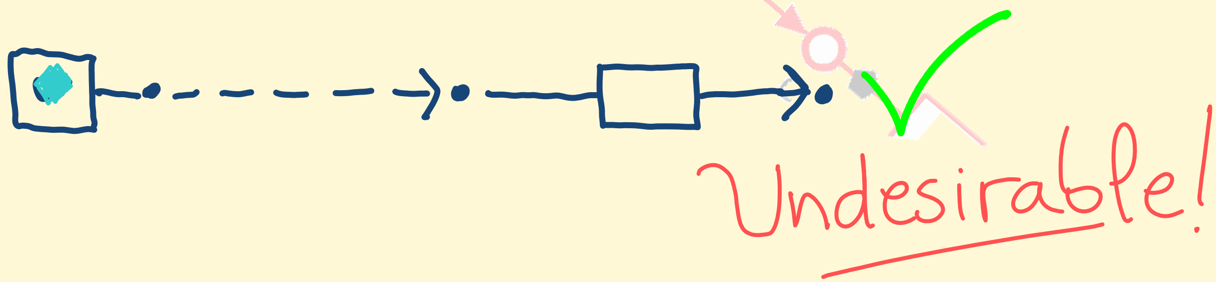


$\{A\}$

The model doesn't indicate which of the two transitions should fire.

# Motivation

- **Cons:**
  - Cannot distinguish between context dependent and context independent behaviour
  - Composition of context dependent behaviour is not according to the informal description of Reo’s behaviour.
  - Not surprisingly reveal not to be appropriated model to constructively characterize the *hiding operation*



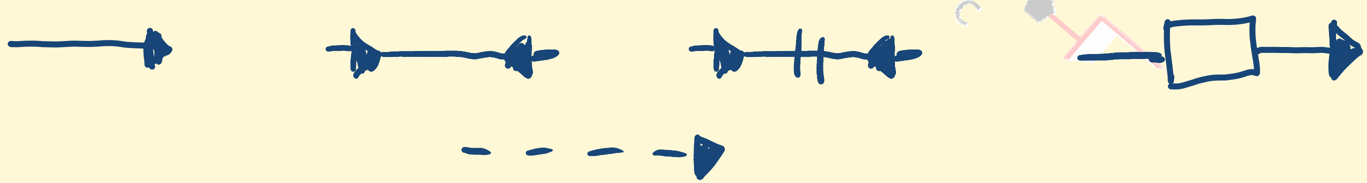
# Reo connectors

- Component **connectors** act as glue code that **connect** and **coordinate** components in component-based systems.
- A component **interacts** with a connector it is connected to **anonymously** and without any knowledge of other components.
- A connector coordinates the concurrent interactions of each of its connected components.
- Complex connectors are built out of an extensible set of channels

# Reo Connectors

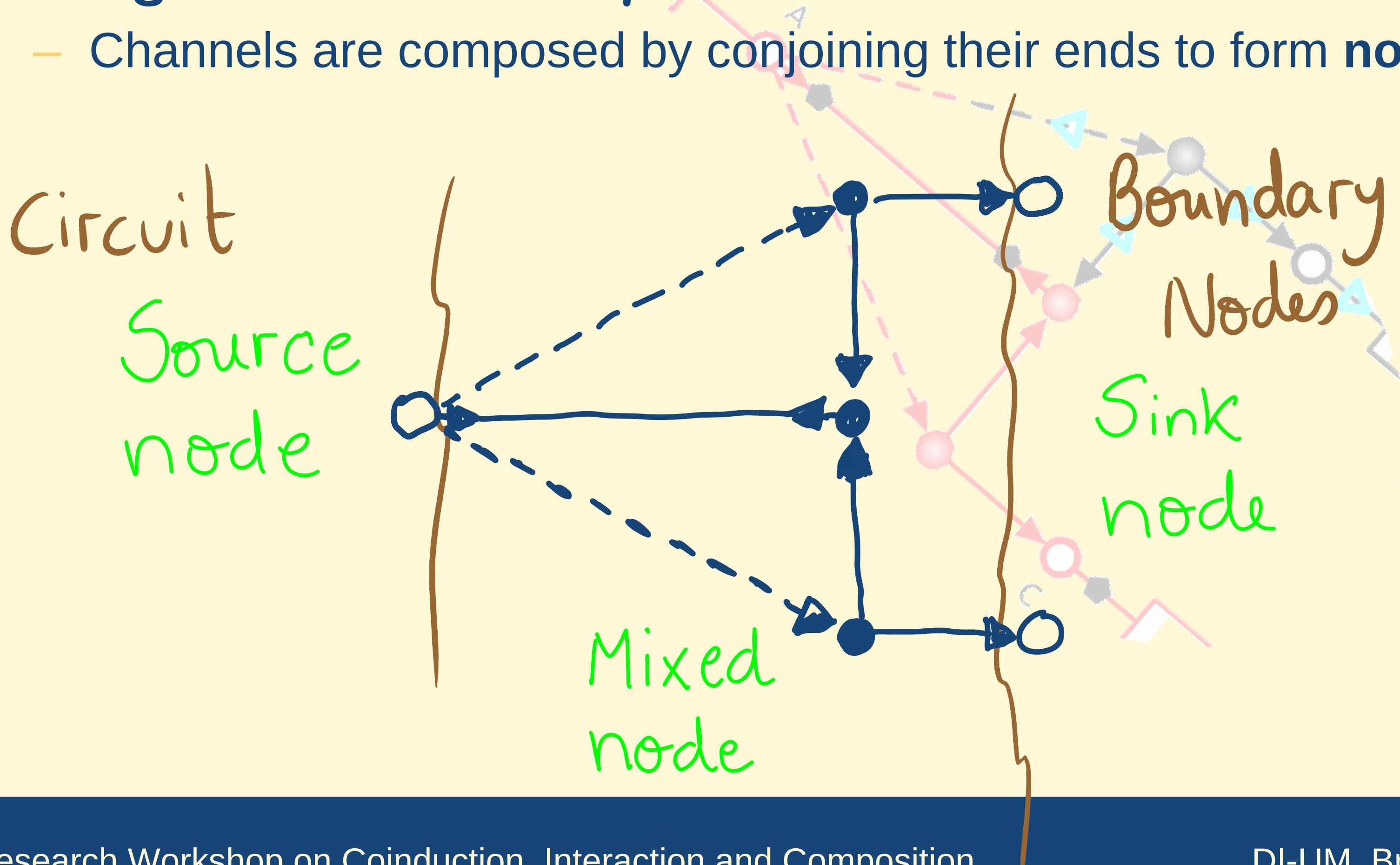
- **Channels**
  - Point to point communication medium with two distinct ends;
  - We have two type of channel ends:
    - **source end** (accepting data)
    - **sink end** (dispensing data)
  - A channel not necessarily need to have both a source end and a sink end;
  - It can instead have two source ends or two sink ends;
  - It has a well defined *behaviour*

Examples:



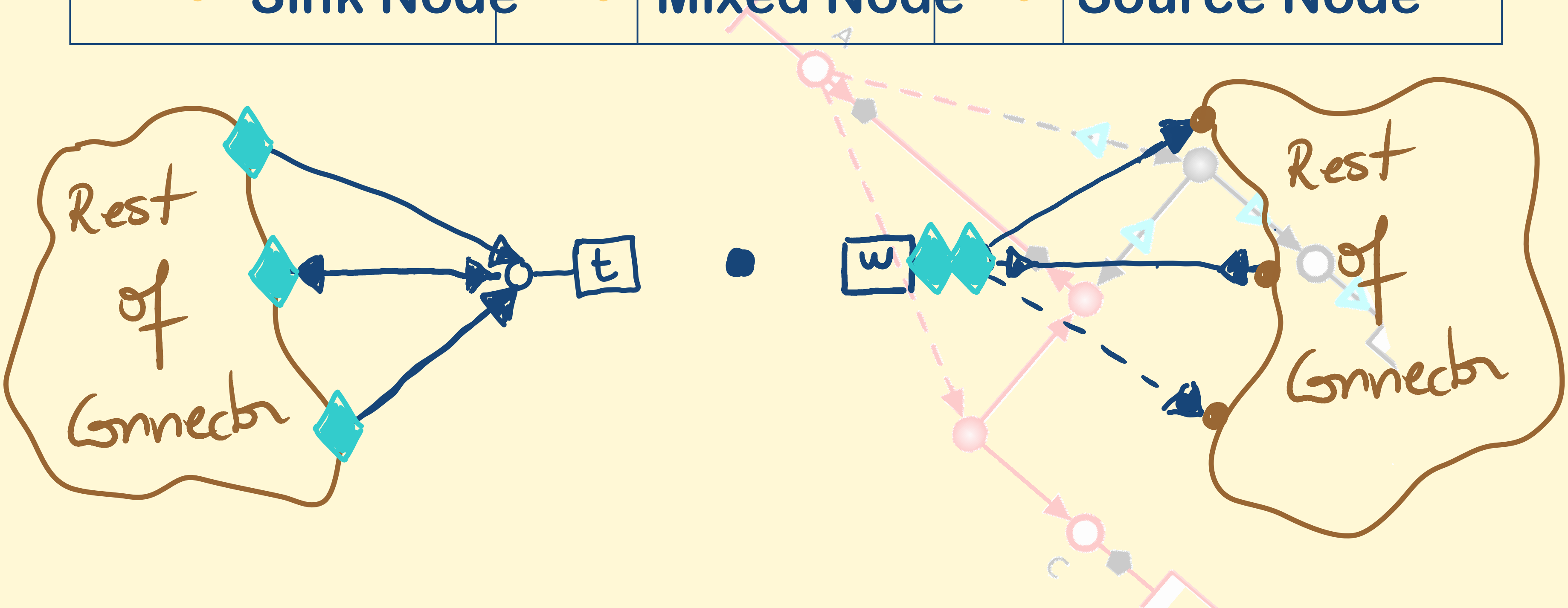
# Reo Connectors

- **Complex connectors are built out of simpler ones using connector composition**
  - Channels are composed by conjoining their ends to form **nodes;**



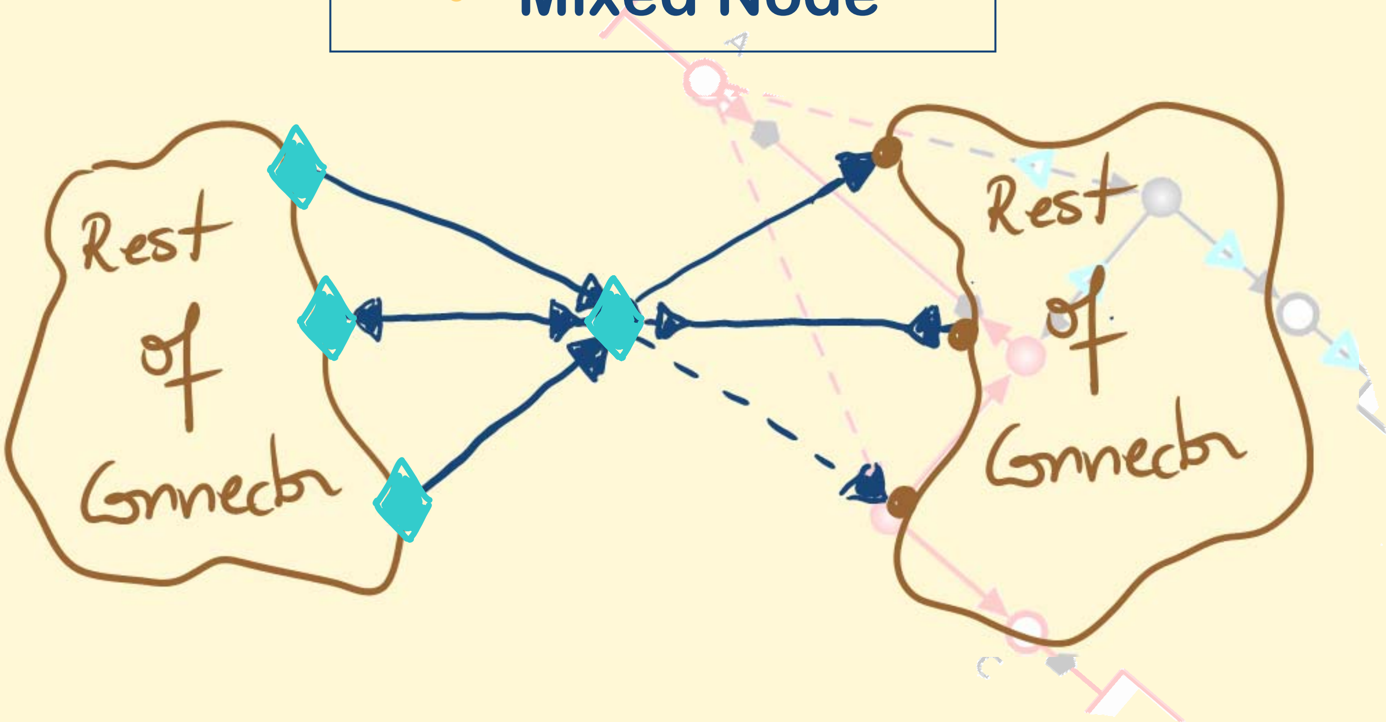
# Reo Connectors - Nodes

- Sink Node
- Mixed Node
- Source Node



# Reo Connectors - Nodes

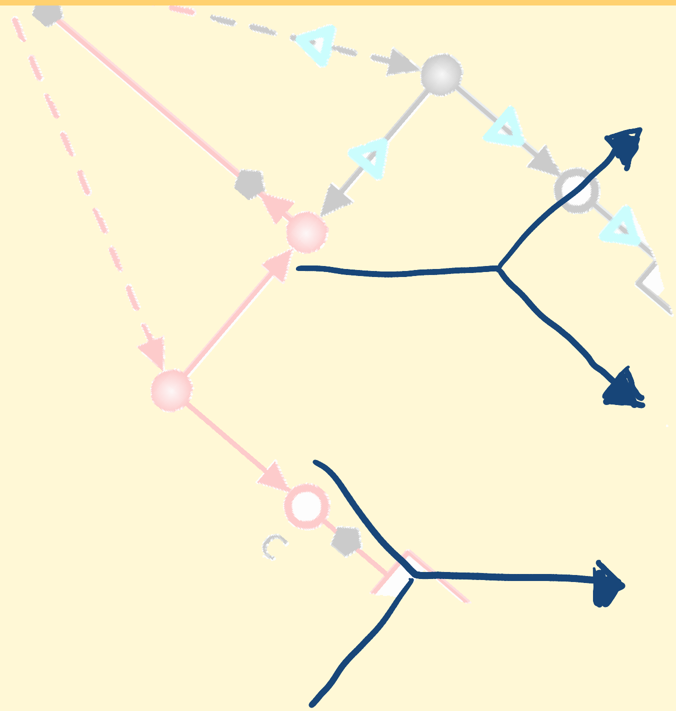
- **Mixed Node**



# Reo Connectors - Nodes

We make the merge and replicate behaviour inherent in Reo nodes explicit modelling them using two additional primitive connectors:
The **Replicator** and the **Merger**.

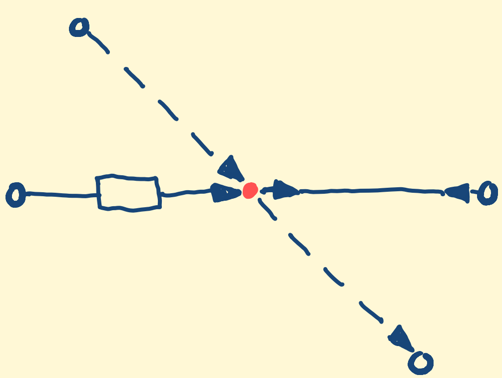
- **Source Nodes**
  - Modelled using the Replicator
- **Sink Nodes**
  - Modelled using the Merger

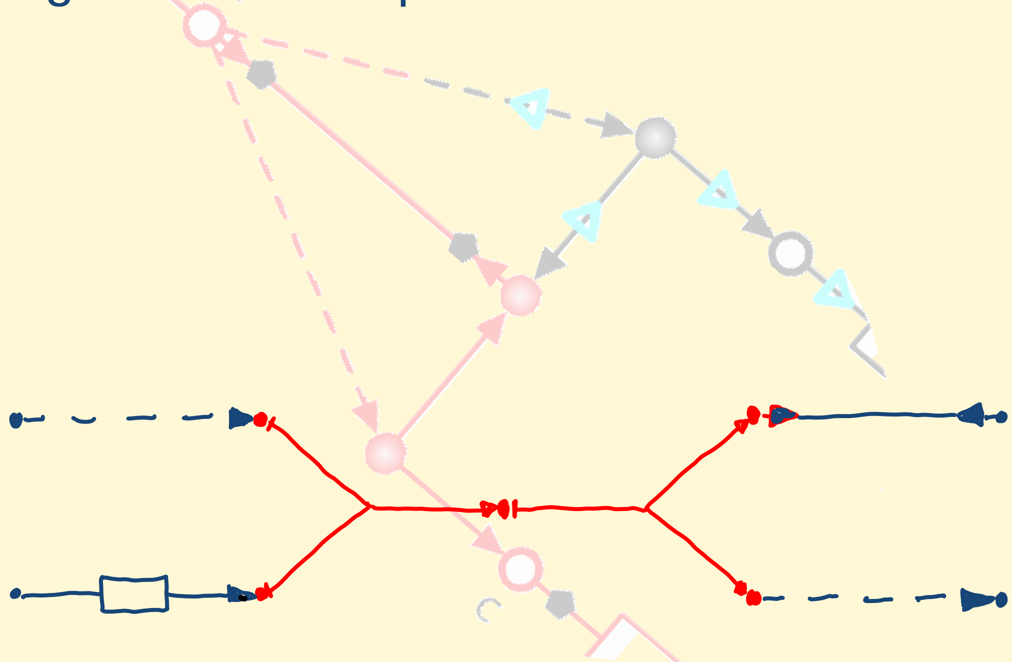


# Modelling decisions

- **Mixed Nodes**
  - Modelled using the Merger and the Replicator

Example:





# Reo Connectors

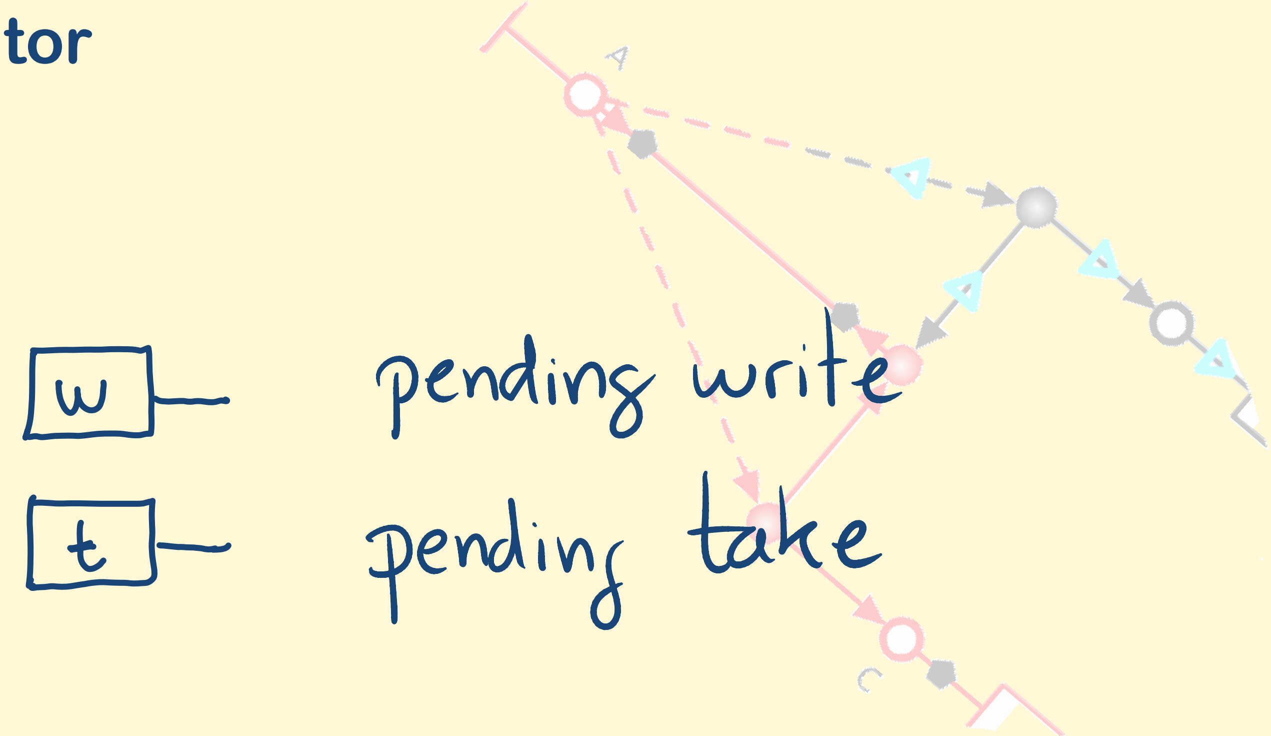
- **I/O operations (write and take)**
  - Components interact with a connector performing I/O operations (**write** and **take**) on *boundary nodes*;
  - A component can perform a **write** with some data on a *source* node or a **take** on a *sink* node.
  - It is by delaying this operations that coordination is achieved;

Note:

We refer to an *I/O operation* that is being *delayed* as a *pending operation*;

# Reo Connectors

- **We consider a pending I/O operation a primitive connector**



# Reo connectors


- **Reo primitive connectors:**
  - Channels
  - Merger and Replicator
  - I/O operation

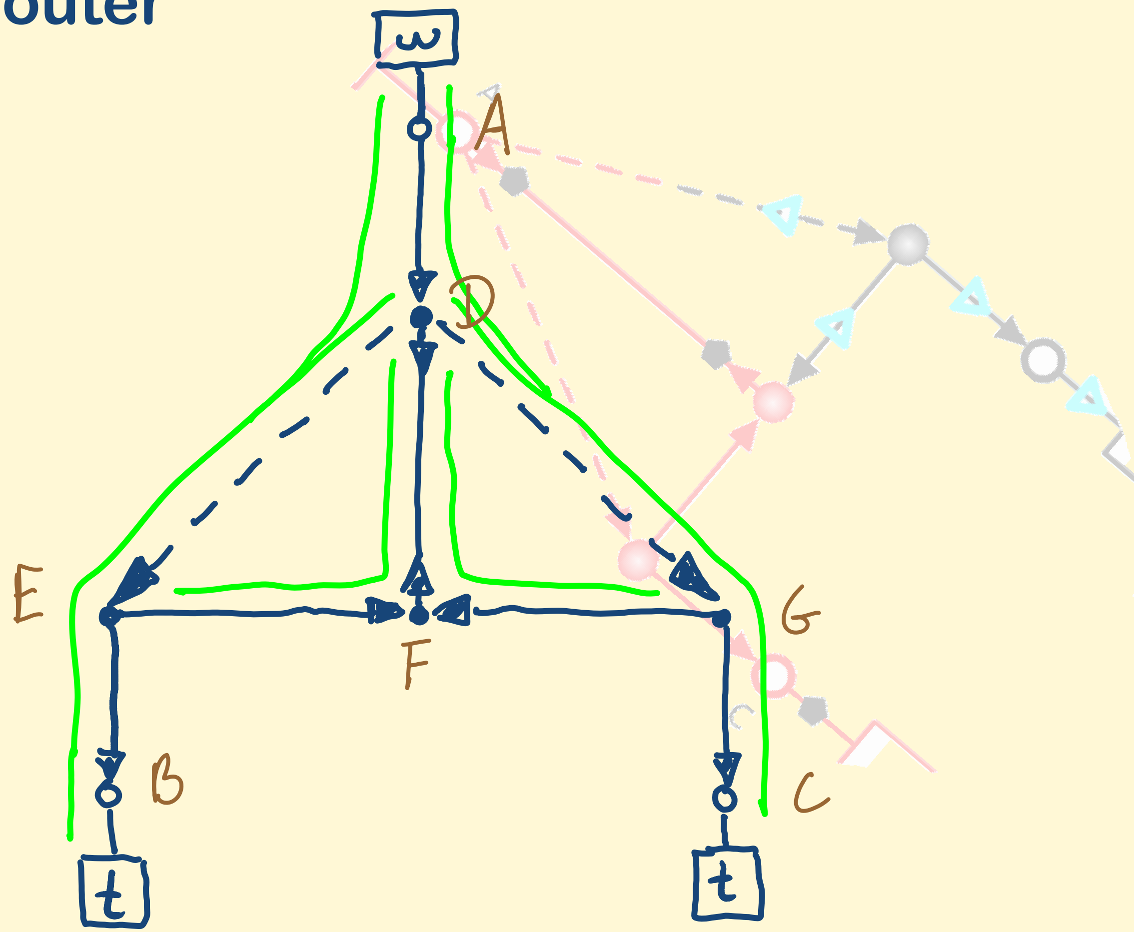
# Colour and Colouring

- Connector Colouring is based on the idea of marking data flow and its absence by *colours* which are elements of a fixed set Colour.

- Each *colouring* of a connector is a solution to the synchronization and exclusion constraints imposed by its channels and nodes. A colouring determines in which part(s) of the connector(s)  data will flow.

# Example

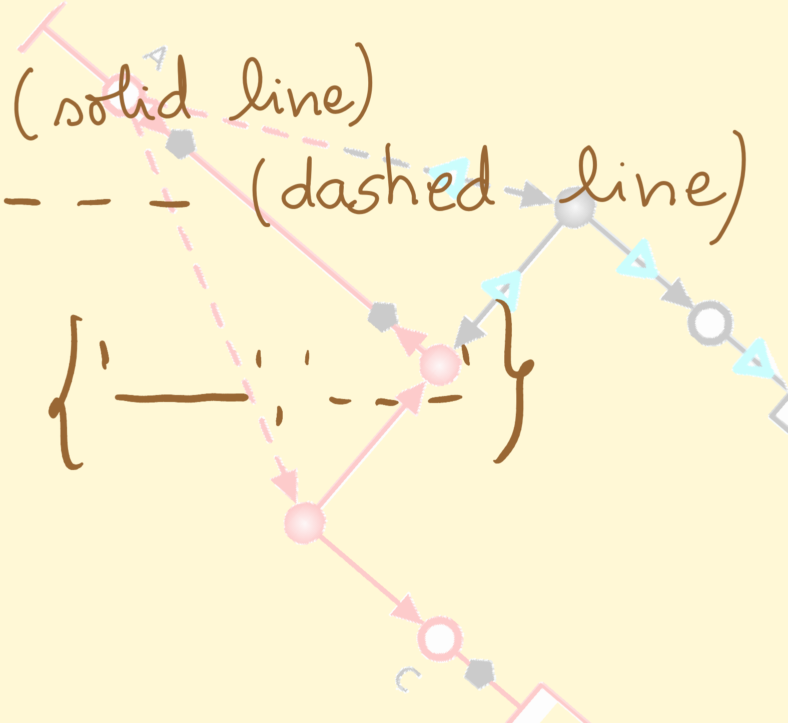
- **Exclusive Router**



# Connector Colouring

- **Let's start to consider our set Colour with two elements:**
  - The colour flow —— (solid line)
  - And the colour no-flow - - - (dashed line)

Colour = {'——', '- - -'}



# Connector Colouring

- Reo semantics dictates that data is never stored or lost at nodes.
- Either data flows out of one end through the node to the other end, or there is no flow at all.
- Thus the data flow at one end attached to a node must be the same as at the other end attached to the node.
- That means that the two ends of a node have always the same colour.

# Connector Colouring

- **Colouring**

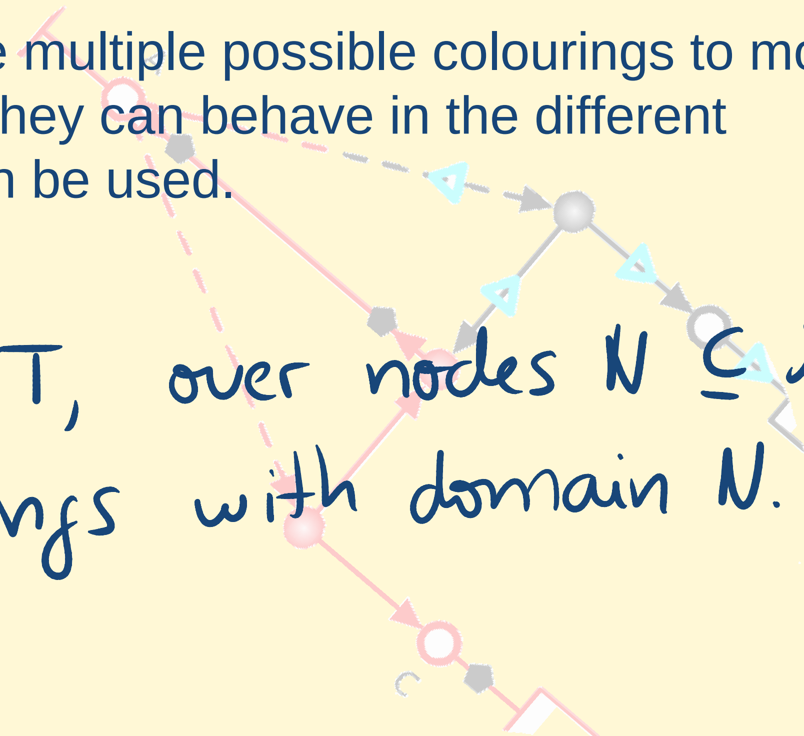
A colouring $\boxed{c : N \to \mathit{Colour}}$ for $N \subseteq \mathit{Node}$ is a function that assigns a colour to every node of a connector.

Remark:

Colouring all the nodes of a connector, in a manner consistent with the colourings of its constituents, produces a valid description of data flow through the connector.

# Connector Colouring

- **Colouring Table**
  - Typically connectors have multiple possible colourings to model the alternative ways that they can behave in the different contexts in which they can be used.



A colouring table, $T$, over nodes $N \subseteq Node$ is a set of colourings with domain $N$.

# Connector Colouring

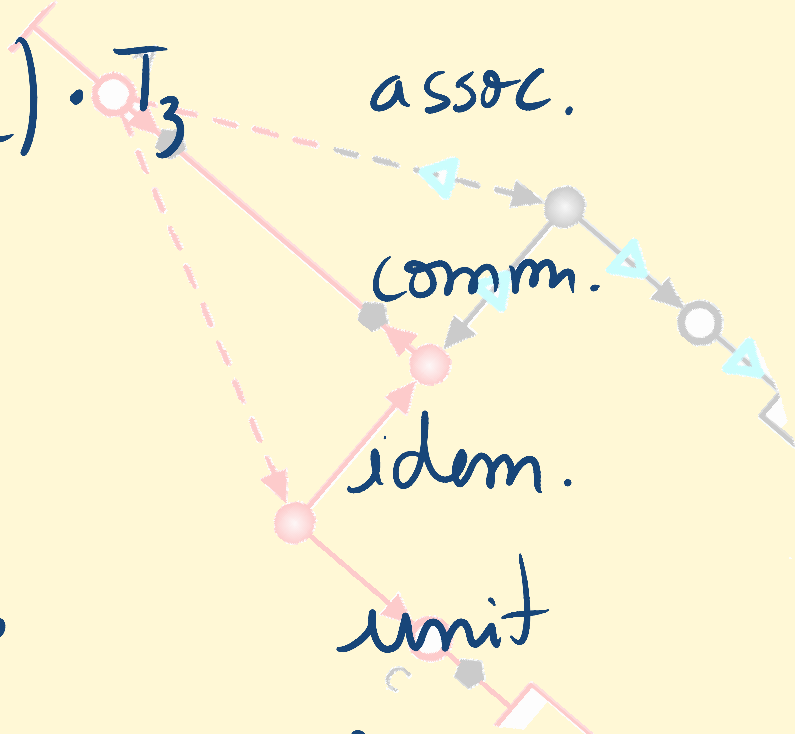
- **Composing Tables**
  - Colouring a connector involves composing the colourings of its constituents so that they agree on the colour of their common nodes.

The join of two tables $T_1$ and $T_2$

$$T_1 \cdot T_2 = \{ \underbrace{c_1 \cup c_2}_{\text{is a function}} \mid c_1 \in T_1,\ c_2 \in T_2,\ n \in dom(c_1) \cap dom(c_2) \Rightarrow c_1(n) = c_2(n) \}$$

# Connector Colouring

- **Properties of the operator •**



$$T_1 \cdot (T_2 \cdot T_3) \equiv (T_1 \cdot T_2) \cdot T_3$$ assoc.

$$T_1 \cdot T_2 \equiv T_2 \cdot T_1$$ comm.

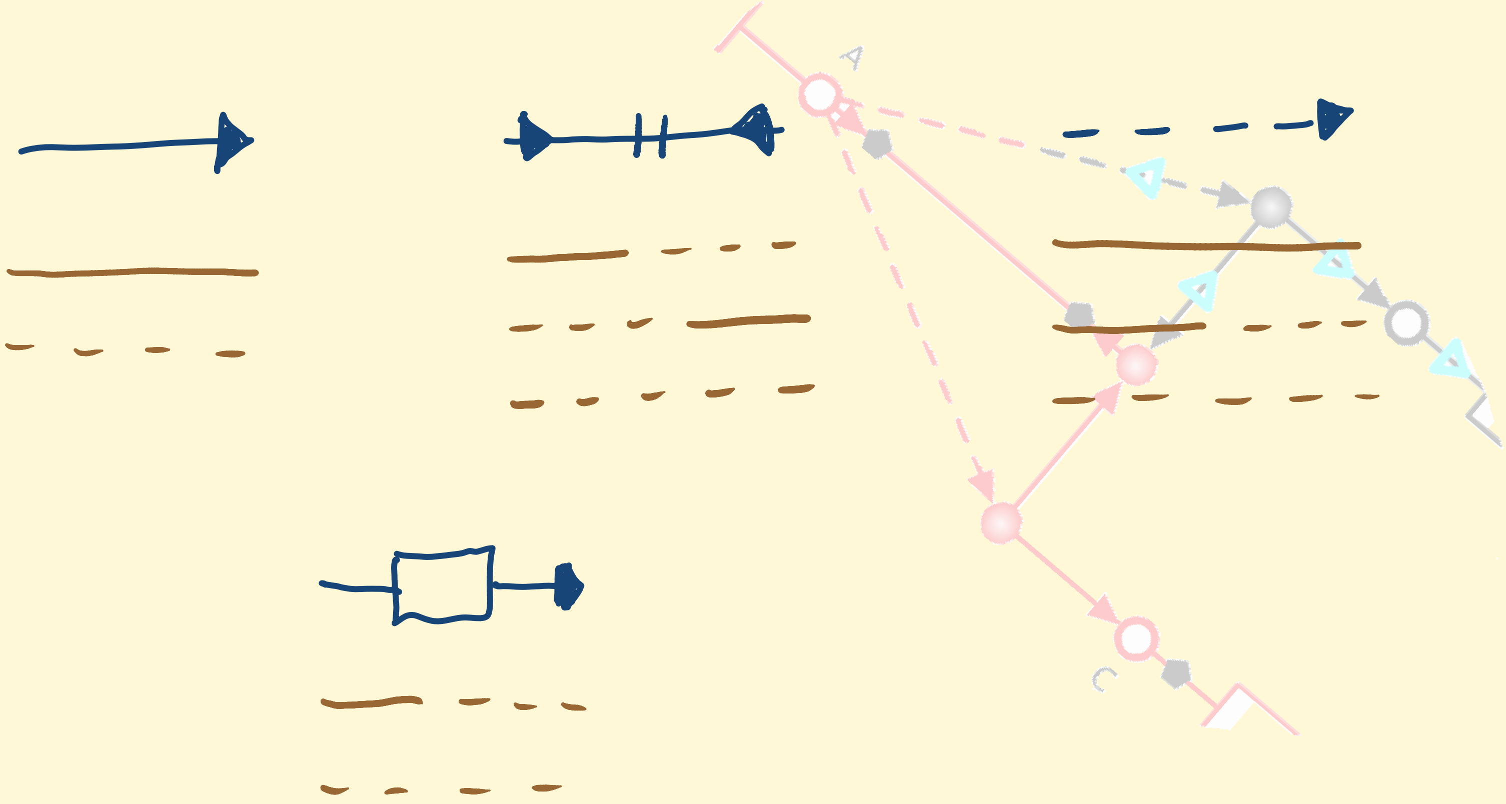
$$T_1 \cdot T_1 \equiv T_1$$ idem.

$$T \cdot 1 \equiv T, \quad 1 = \{\emptyset\}$$ unit

$$T \cdot 0 \equiv 0, \quad 0 = \emptyset$$ zero

# Connector Colouring

- Channels



# Connector Colouring

- **I/O operations**

- **Replicator**

# Connector Colouring

- **Merger**

# Connector Colouring

- **Exercise**

# Connector Colouring

- We exchange the colour no-flow with two distinct colours.

# Connector Colouring

- **Determines the behaviour of Reo Connectors**
  - resolving its context dependent synchronization and mutual exclusion constraints;
- **Colouring a Reo connector consists of**
  - In a specific (connector) state
  - with given boundary conditions (I/O requests)
  - calculate the routing alternatives for data flow.

# Summary

- **Connector Colouring improves on the existing models in two major aspects:**
  - It distinguishes the alternatives of behaviour of a connector at a finer level of granularity by modelling accurately context dependent behaviour;
    - Turning it into the closest to the informal description of Reo's behaviour present in the original Reo paper;
  - The composition operator has a number of formal properties (assoc., comm. Idem.) that make it quite suitable for a distributed implementation.
    - Does not require backtracking nor history computations;
    - Concurrent parties can combine each other's partially computed results;

# Conclusion

- Have presented a model based on connector colouring for resolving the context dependent synchronization and mutual exclusion constraints required to determine the routing for dataflow in Reo connectors.

- Gave hints on how it facilitates the dataflow computation and implementation of Reo connectors in a distributed computing environment.

# Ongoing Work

- Connector Colouring I – Synchronisation and Context Dependency
  – 2-colour model and 3-colour model
- **Connector Colouring II – Constructive 3-colouring**
  – Identify and disallow non-constructive colourings arising in the presence of synchronous loops.
- **Connector Colouring III – Hidden machine**
  – Addressing Reo’s hide operation
- **Parallel:**
  – Animation of Reo Connectors
  – Distributed implementation of Reo with connector colouring as the underlying semantics