Transfer Request Broker: Resolving Input-Output Choice

Oliver Faust, Bernhard H.C. Sputh, Alastair R. Allen

University of Aberdeen

September 8, 2008

Motivation

- **Problem:** resolving input and output choice
  - Know the network state
  - Store the network state
  - Update the network state
- **Solution:** Transfer Request Broker (TRB)
  - Relation matrix, an efficient way to store and update the network state
  - Compact representation
  - The size of the relation matrix is known during design time $\longrightarrow$ no infinite buffers required
- **Realisation:** formal model
  - Matrix operation support for CSPM
  - Classical CSP model for specification and implementation

Agenda

This session is structured as follows:

- Problem specification
  - Problem statement
  - CSP *SPECIFICATION* model
- Refinement from specification to implementation
  - Matrix based network topology
  - CSP *IMPLEMENTATION* model
- Discussion of the CSP models
  - Sequential nature of the search algorithm.
  - Model checking.
- Conclusions

Water Risk Management Europe

The project was sponsored by the EC:

- EC FP6 IST - Water Risk Management EuRope (WARMER)
- EC no. 034472 FP6-2005-IST-5

The aims of the WARMER project are:

1. Sensor development;
2. In-situ Monitoring Station development;
3. In-situ Sensing Data Collection and Presentation;
4. Remote Sensing Data Collection and Presentation;
5. Fusion and Presentation of In-situ and Remote Sensing Data.

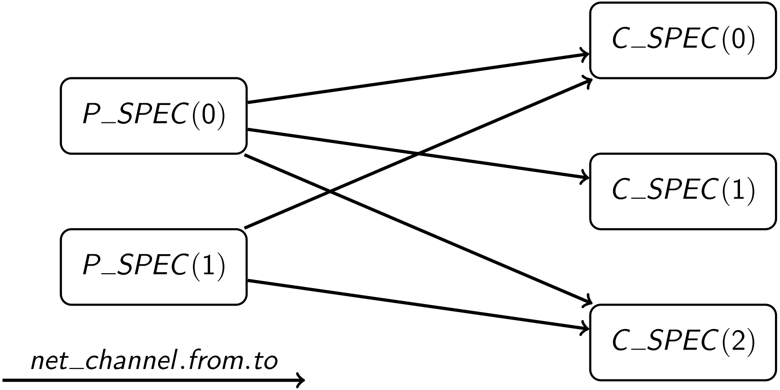
Problem statement

Two conflicting facts:

- The CSP process algebra explicitly allows resolving input and output guards.
  - Symmetry
  - Choice over input and output ensures that every parallel command can be translated into a sequential equivalent.
- Programming languages which offer CSP primitives resolve only input choice (alternation).
  - Computational complexity
  - Code size

Refine a system which uses input and output choice into a system which uses only input choice.

SPECIFICATION process network



CSP model

Define the individual processes:

$$P\_SPEC(i) = in.i?x \rightarrow \Box_{j \in \mathsf{p\_set}(i)} \underbrace{net\_channel.i.j!x}_{\text{output guards}} \rightarrow P\_SPEC(i)$$

$$C\_SPEC(j) = \Box_{i \in \mathsf{c\_set}(j)} \underbrace{net\_channel.i.j?x}_{\text{input guards}} \rightarrow out.j!x \rightarrow C\_SPEC(j)$$

Define producer and consumer groups:

$$PRODUCER\_SPEC = |||_{i \in \{0..n-1\}} P\_SPEC(i)$$

$$CONSUMER\_SPEC = |||_{j \in \{0..m-1\}} C\_SPEC(j)$$

Make producer and consumer communicate over the *net_channel*s:

$$SPECIFICATION = CONSUMER\_SPEC \underset{\{|net\_channel|\}}{\|} PRODUCER\_SPEC$$

Link signals

1. $net\_channel.0.0$ connects $P\_SPEC(0)$ to $C\_SPEC(0)$;
2. $net\_channel.0.1$ connects $P\_SPEC(0)$ to $C\_SPEC(1)$;
3. $net\_channel.0.2$ connects $P\_SPEC(0)$ to $C\_SPEC(2)$;
4. $net\_channel.1.0$ connects $P\_SPEC(1)$ to $C\_SPEC(0)$;
5. $net\_channel.1.2$ connects $P\_SPEC(1)$ to $C\_SPEC(2)$.

Relation matrix

	$C\_SPEC(0)$	$C\_SPEC(1)$	$C\_SPEC(2)$
$P\_SPEC(0)$	1	1	1
$P\_SPEC(1)$	1	0	1

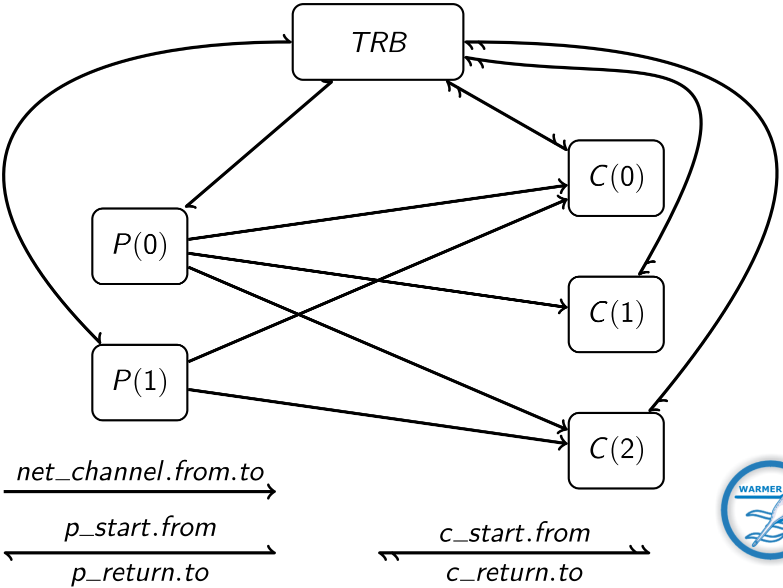
Solution

One independent entity which resolves input and output choice.

This entity must have the following properties:

- It needs to know (get informed) about the network state.
- Efficiency of the choice resolution algorithm.
- Efficiency in storing and updating the network state.
- It needs to communicate the choice result.

IMPLEMENTATION process network



Example

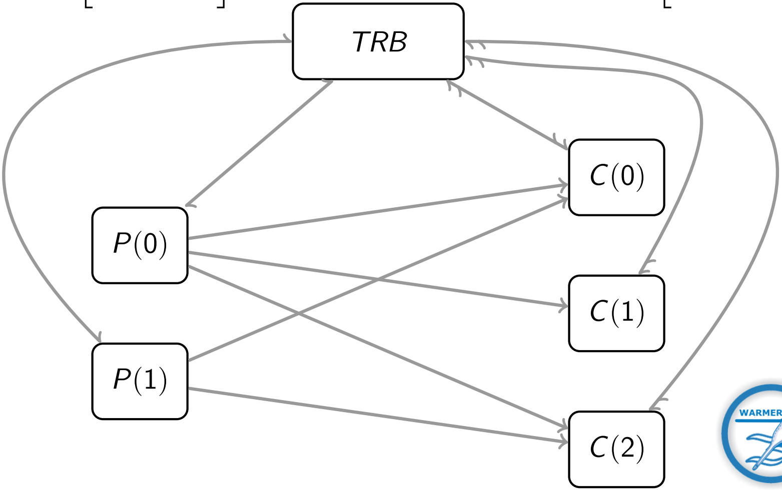
The mathematics are in the paper and not in the presentation.

This motto leads to a visual example which explains the TRB functionality. The following list sets the goals for the example:

- The individual channel transactions are shown.
- The change of the relation matrix in response to these transactions is shown.

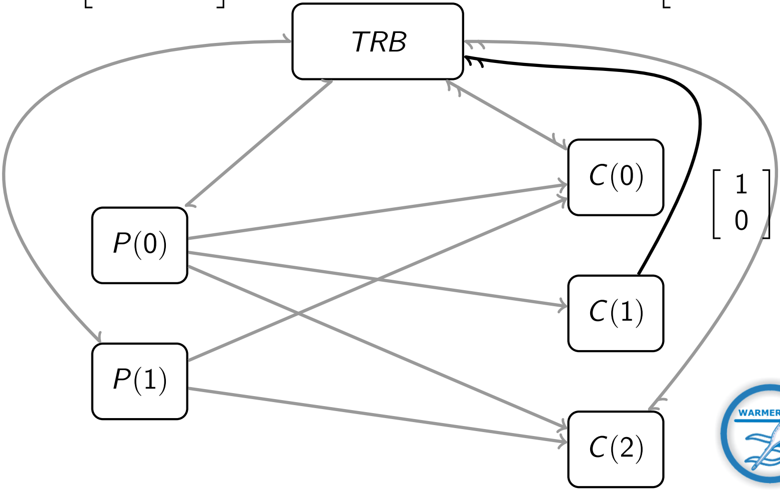
Example: initial setup

$$p\_array = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad c\_array = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$



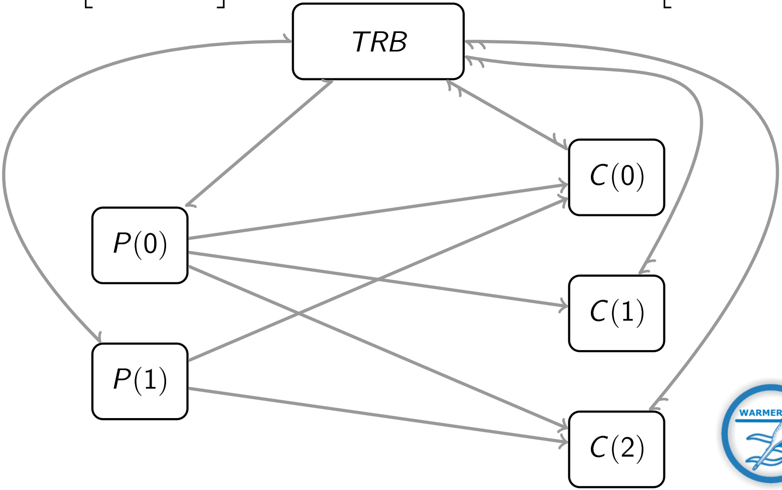
Example: $C(1)$ communicates with the TRB

$$p\_array = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad c\_array = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

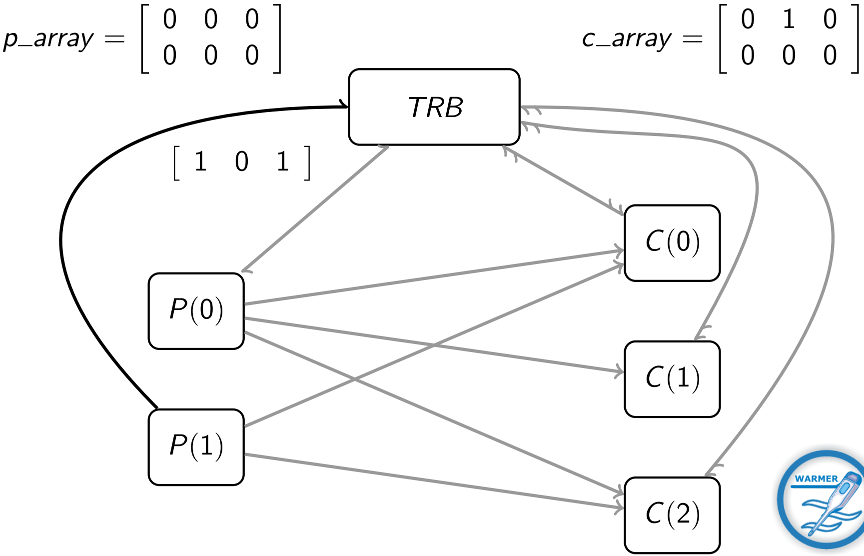


Example: update *c_array*

$$p\_array = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad c\_array = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

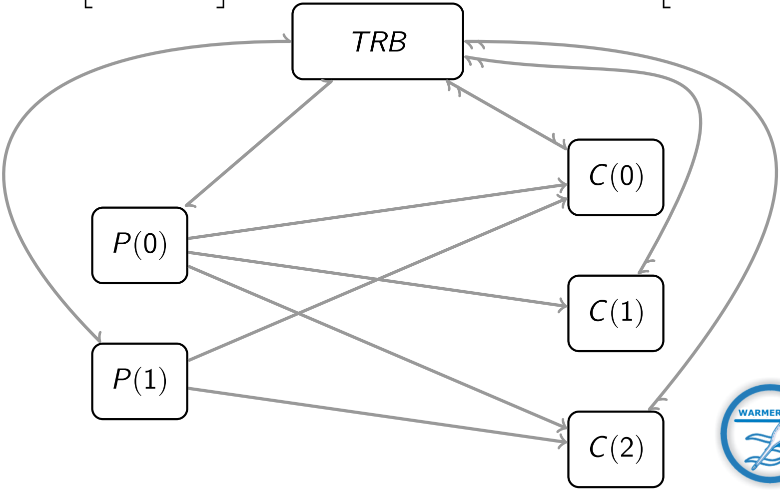


Example: $P(1)$ communicates with the TRB



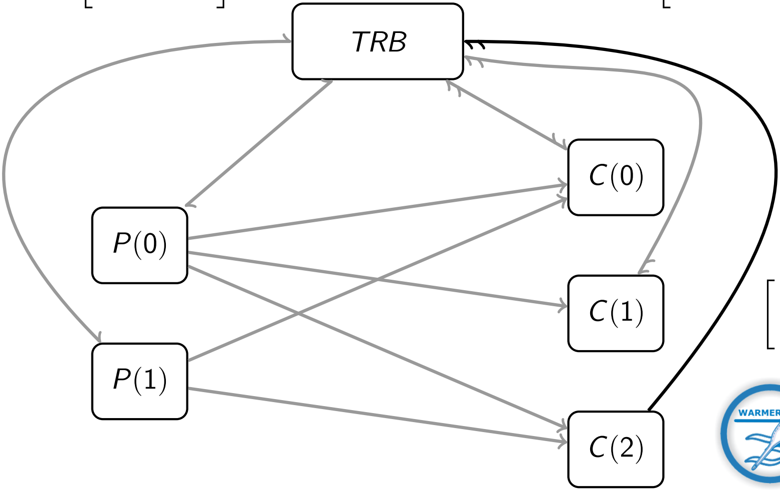
Example: update *p_array*

$$p\_array = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \end{bmatrix} \qquad c\_array = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$



Example: $C(2)$ communicates with the *TRB*

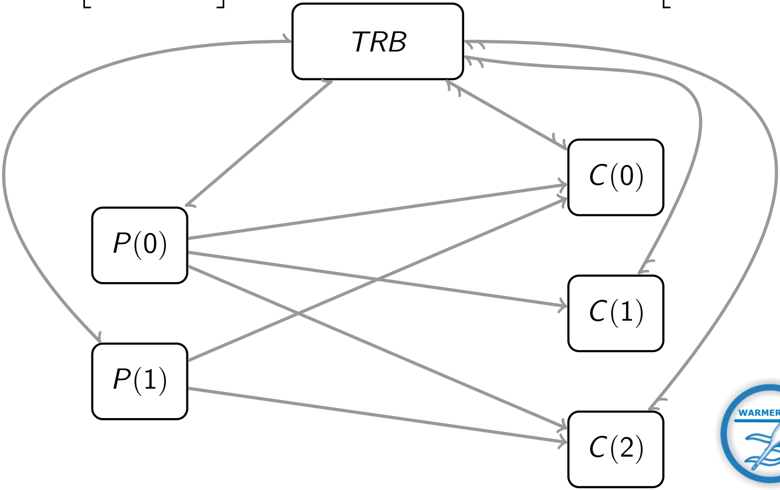
$$p\_array = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \end{bmatrix} \qquad c\_array = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$



$$\begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

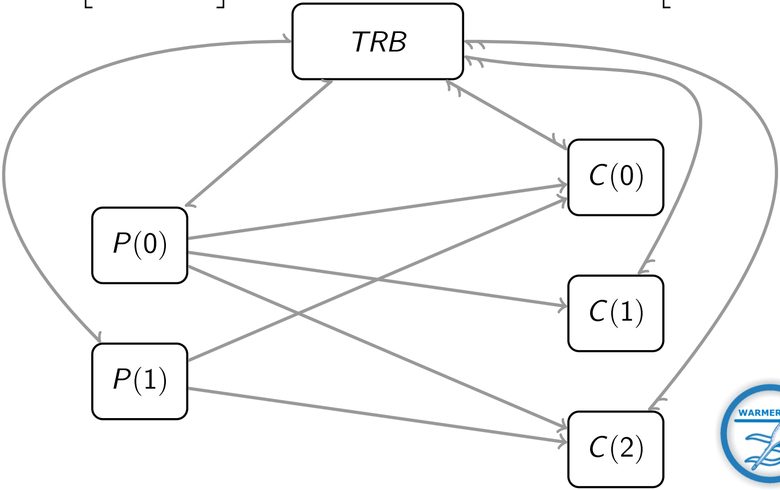
Example: update *c_array*

$$p\_array = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & \mathbf{1} \end{bmatrix} \qquad c\_array = \begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & \mathbf{1} \end{bmatrix}$$



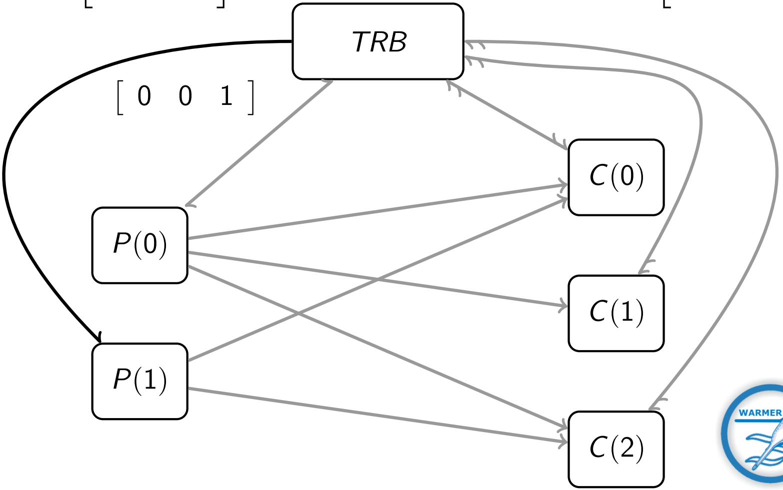
Example: reset $P(1)$ row vector in p_array

$$p\_array = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad c\_array = \begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$



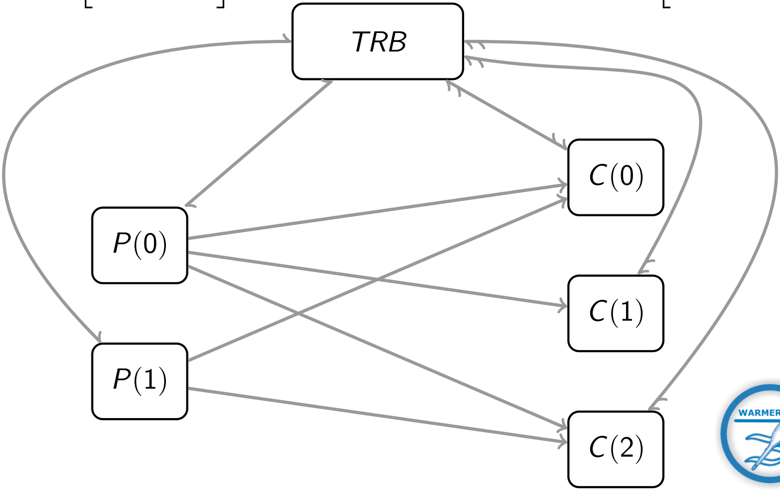
Example: the *TRB* communicates with $P(1)$

$$p\_array = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad c\_array = \begin{bmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix}$$



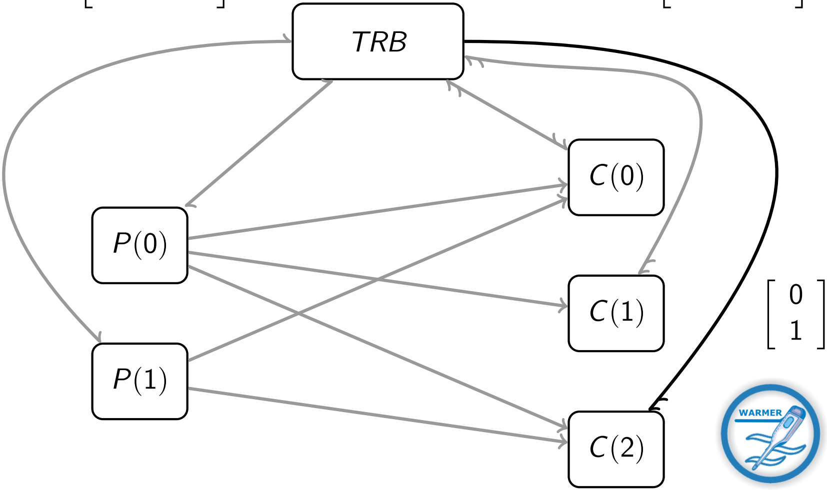
Example: reset $C(2)$ column vector in c_array

$$p\_array = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad c\_array = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$



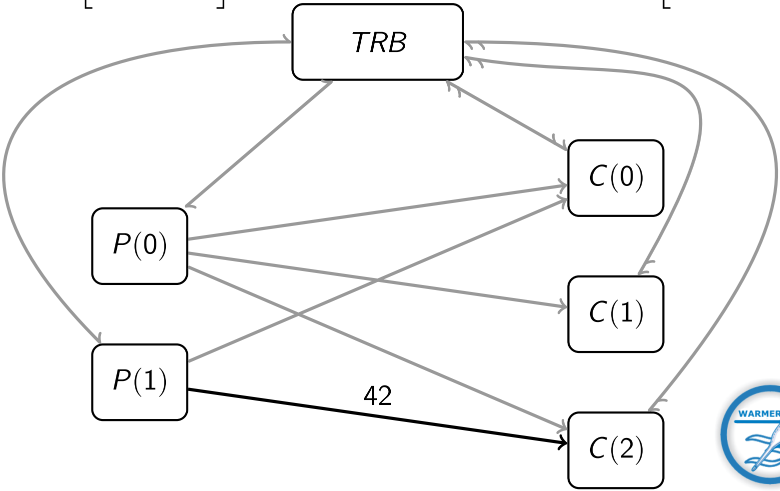
Example: the *TRB* communicates with $C(2)$

$$p\_array = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad c\_array = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$



Example: data transfer

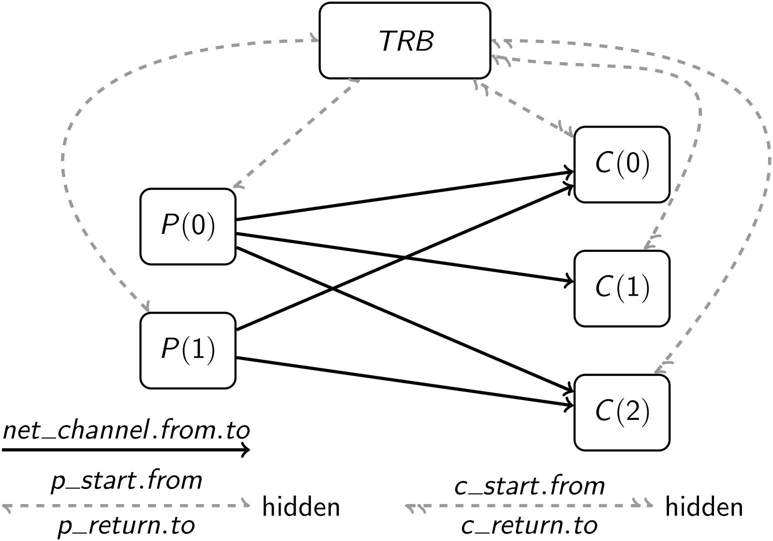
$$p\_array = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \qquad c\_array = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$



Model checking

- Setup:
  - *SPECIFICATION* model
  - *IMPLEMENTATION* model
- Checks:
  - Deadlock
  - Divergence
  - Deterministic
  - Trace refinement

IMPLEMENTATION process network with the communication to and from the *TRB* hidden



Model checking results

FDR output:

- ✓ SPECIFICATION deadlock free [F]
- ✓ SPECIFICATION livelock free
- ✓ IMPLEMENTATION deadlock free [F]
- ✓ IMPLEMENTATION livelock free
- ✓ SPECIFICATION deterministic [FD]
- ✗ IMPLEMENTATION deterministic [FD]
- ✓ IMP deterministic [FD]
- ✓ SPECIFICATION [T= IMPLEMENTATION
- ✓ IMPLEMENTATION [T= SPECIFICATION

Absent checks:

- Failure refinement
- Failure divergence refinement

Conclusions

Summary:

- **Problem:** resolving input and output choice
- **Solution:** Transfer Request Broker (TRB)
- **Realisation:** formal model

Main ideas presented:

- An external / independent which controls the network.
- Represent the network with a relation matrix.
- Extend CSP_M with matrix operations.

Further work

There are only two points for future work:

1. Scalability:
    - Remove the single point of failure.
    - Remove the bottle neck.
2. Priority
3. Mobility

Question and Answers

- An external / independent which controls the network.
- Represent the network with a relation matrix.
- Extend CSP_M with matrix operations.