

Reo: Operational Semantics, Animation and Model Checking

M. Mousavi¹, M. Sirjani^{2,3}, F. Arbab²

¹Eindhoven University of Technology, ²CWI, Amsterdam

³Sharif University of Technology

OAS Colloquium, March 2004,
TU/e, Eindhoven

Overview

- [An \(Informal\) Introduction to Reo](#)
- 2. Operational Semantics
- 3. From Reo to Maude
- 4. Thoughts on Components and Mobility (May Skip This!)
- 5. Conclusion and Future Perspective

An (Informal) Introduction to Reo

Components:

1. Units of **independent functionality** and deployment
2. Encapsulated (**blackbox**) with well defined **interface**

An (Informal) Introduction to Reo

Components:

1. Units of **independent functionality** and deployment
2. Encapsulated (**blackbox**) with well defined **interface**

Conclusions:

1. **Components** should be **composed** to form an **application**.
2. **Composition** mechanism is different from components.

Glue code: usually swept under the carpet.

An (Informal) Introduction to Reo

Reo:

1. **Coordination** language for composing components;
2. Electronic circuits metaphor: take **basic elements**, **plug**, and play!;
3. Support for dynamic **reconfigurations**, **mobility** (other buzz words!...)

An (Informal) Introduction to Reo

Glossary of Terms:

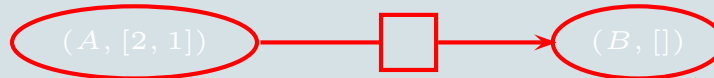
- Basic types of elements: **Basic connector types**



An (Informal) Introduction to Reo

Glossary of Terms:

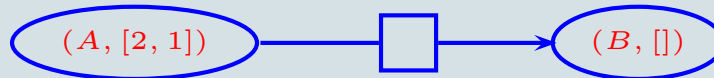
- Basic types of elements: **Basic connector types**
- Instances of basic connectors: **Channel instances**
- Channel instances connect **two sets** of **nodes**



An (Informal) Introduction to Reo

Glossary of Terms:

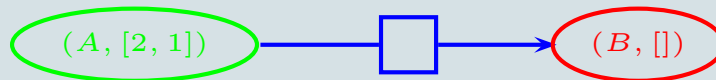
- Basic types of elements: **Basic connector types**
- Instances of basic connectors: **Channel instances**
- Channel instances connect **two sets** of **nodes**
- Nodes are pairs of **location** and **data** value



An (Informal) Introduction to Reo

Glossary of Terms:

- Basic types of elements: **Basic connector types**
- Instances of basic connectors: **Channel instances**
- Channel instances connect **two sets** of **nodes**
- Nodes are pairs of **location** and **data** value
- Node sets can be **source** or **sink** sets



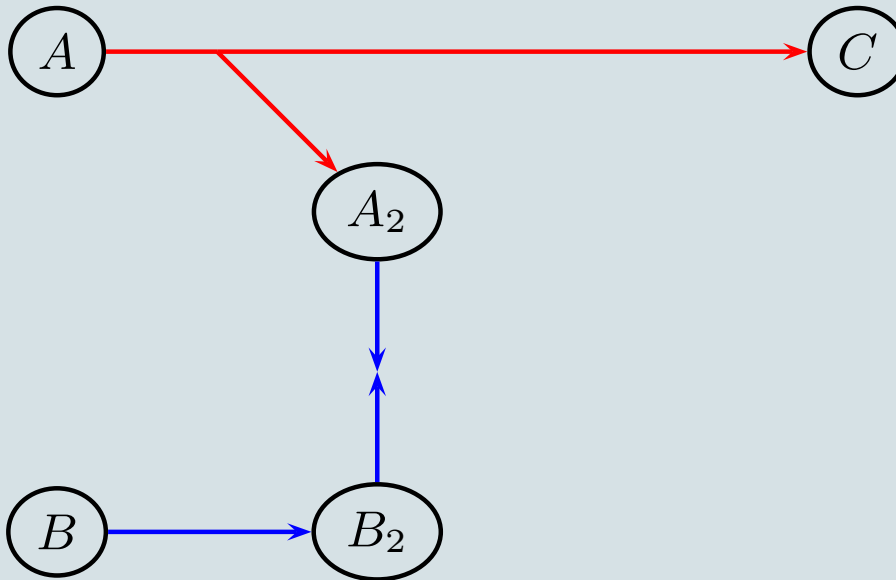
An (Informal) Introduction to Reo

Basic Connector Types:

- *Synchronous* \mapsto
- *Synchronous drain* $\succ\prec$
- *Synchronous lossy* $--\rightarrow$
- *One place fifo* $-\square\rightarrow, -a\rightarrow$
- *Infinite fifo* $-[\]\rightarrow -[u]\rightarrow.$
- *Fork* \leftarrow
- *Merge* \rightarrow

An (Informal) Introduction to Reo

A Small Example:



I. An (Informal) Introduction to Reo

$BChannel ::= \langle NodeSet\ BConnector\ NodeSet \rangle$

$Node ::= (Name, DataSeq)$

$BConnector ::= \mapsto \mid \triangleright \triangleleft \mid \dashrightarrow \mid$
 $\neg \square \rightarrow \mid \neg a \rightarrow \mid \neg [] \rightarrow \mid \neg [u] \rightarrow \mid \preceq \mid \succeq$

I. An (Informal) Introduction to Reo

Constraints on the Abstract Syntax:

I. Source and Sink **Cardinalities**:

(a) $1to1$: \mapsto , $-\!\!\rightarrow$, $-\square\!\!\rightarrow$, $-\lceil \rceil\!\!\rightarrow$

(b) $2to0$: $\succ\prec$

(c) $1ton$: $\!-\prec$

(d) $nto1$: $\succ\!-\$

I. An (Informal) Introduction to Reo

Constraints on the Abstract Syntax:

1. Source and Sink **Cardinalities**:

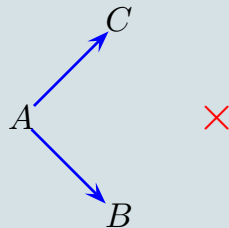
(a) $1to1$: \mapsto , \dashrightarrow , \dashrightarrow , \dashrightarrow

(b) $2to0$: $\succ\prec$

(c) $1ton$: \dashleftarrow

(d) $nto1$: $\succ\text{---}$

2. **Plugging** Principle



I. An (Informal) Introduction to Reo

Constraints on the Abstract Syntax:

1. Source and Sink **Cardinalities**:

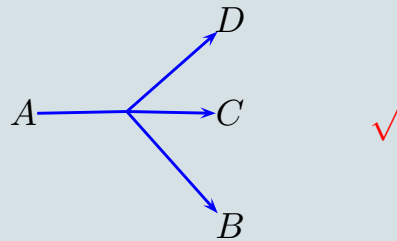
(a) $1to1$: \mapsto , \dashrightarrow , \dashrightarrow , \dashrightarrow

(b) $2to0$: $\succ\prec$

(c) $1ton$: \dashleftarrow

(d) $nto1$: $\succ\text{---}$

2. **Plugging** Principle



I. An (Informal) Introduction to Reo

Constraints on the Abstract Syntax:

1. Source and Sink **Cardinalities**:

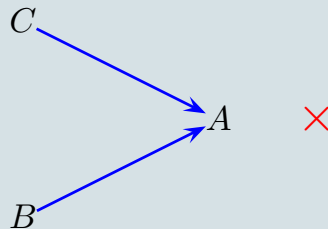
(a) $1to1$: \mapsto , \dashrightarrow , $\dashrightarrow\Box$, $\dashrightarrow\lceil$

(b) $2to0$: $\succ\prec$

(c) $1ton$: \dashleftarrow

(d) $nto1$: $\succ\text{---}$

2. **Plugging** Principle



I. An (Informal) Introduction to Reo

Constraints on the Abstract Syntax:

1. Source and Sink Cardinalities:

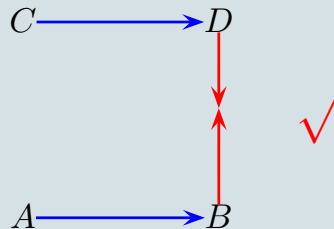
(a) $1to1$: \mapsto , \dashrightarrow , $\dashv\!\!\!\dashv\rightarrow$, $\dashv\!\!\!\dashv\!\!\!\rightarrow$

(b) $2to0$: $\succ\prec$

(c) $1ton$: \dashleftarrow

(d) $nto1$: $\supset\!-\!$

2. **Plugging** Principle



I. An (Informal) Introduction to Reo

Constraints on the Abstract Syntax:

1. Source and Sink **Cardinalities**:

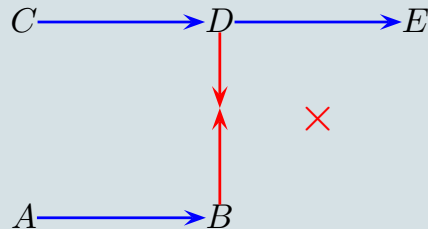
(a) *1to1*: \mapsto , \dashrightarrow , $\dashrightarrow\Box$, $\dashrightarrow\lceil$

(b) *2to0*: $\succ\prec$

(c) *1ton*: \dashleftarrow

(d) *nto1*: $\succ\text{---}$

2. **Plugging** Principle



I. An (Informal) Introduction to Reo

Constraints on the Abstract Syntax:

I. Source and Sink **Cardinalities**:

(a) $1to1$: \mapsto , \dashrightarrow , $\dashrightarrow\Box$, $\dashrightarrow\lceil$

(b) $2to0$: $\succ\prec$

(c) $1ton$: \dashleftarrow

(d) $nto1$: $\succ\text{---}$

2. **Plugging** Principle

3. **Congestion Freedom**

4. Source / Sink / **Hidden** Node Sets

Overview

✓ An (Informal) Introduction to Reo

→ [Operational Semantics](#)

3. From Reo to Maude

4. Thoughts on Components and Mobility

5. Conclusion and Future Perspective

Operational Semantics

Honorable Mentions:

- F. Arbab, Farhad and J.J.M.M. Rutten, A [coinductive calculus](#) of component connectors, Proceedings of WADT'02;
- J.J.M.M. Rutten, Elements of [stream calculus](#), Proceedings of MFPS'01;
- F. Arbab, C. Baier, J.J.M.M. Rutten and M. Sirjani, Modeling component connectors in Reo by [constraint automata](#), Proceedings of FLOCASA'03.

Operational Semantics

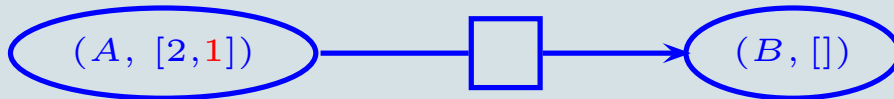
$$\text{(OFifo)} \frac{}{\langle A \text{--}\square\text{--}\rightarrow B, \{A \mapsto u \frown d\} \uplus \sigma \rangle \Rightarrow \langle A \text{--}d\text{--}\rightarrow B, \{A \mapsto u\} \uplus \sigma \rangle}$$

$$\text{(OFifo)} \frac{}{\langle A \text{--}d\text{--}\rightarrow B, \{B \mapsto u\} \uplus \sigma \rangle \Rightarrow \langle A \text{--}\square\text{--}\rightarrow B, \{B \mapsto d \frown u\} \uplus \sigma \rangle}$$

Operational Semantics

$$\text{(OFifo)} \frac{}{\langle A \text{--}\square\text{--}\rightarrow B, \{A \mapsto u \wedge d\} \uplus \sigma \rangle \Rightarrow \langle A \text{--}d\text{--}\rightarrow B, \{A \mapsto u\} \uplus \sigma \rangle}$$

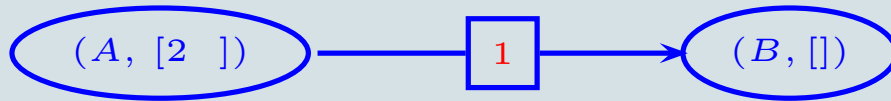
$$\text{(OFifo)} \frac{}{\langle A \text{--}d\text{--}\rightarrow B, \{B \mapsto u\} \uplus \sigma \rangle \Rightarrow \langle A \text{--}\square\text{--}\rightarrow B, \{B \mapsto d \wedge u\} \uplus \sigma \rangle}$$



Operational Semantics

$$\text{(OFifo)} \frac{}{\langle A \text{---}\square\text{---} B, \{A \mapsto u \frown d\} \uplus \sigma \rangle \Rightarrow \langle A \text{---}d\text{---} B, \{A \mapsto u\} \uplus \sigma \rangle}$$

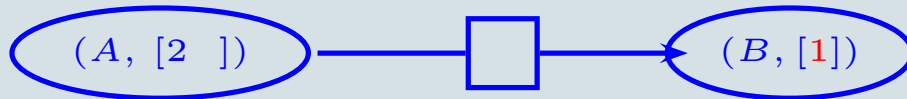
$$\text{(OFifo1)} \frac{}{\langle A \text{---}d\text{---} B, \{B \mapsto u\} \uplus \sigma \rangle \Rightarrow \langle A \text{---}\square\text{---} B, \{B \mapsto d \frown u\} \uplus \sigma \rangle}$$



Operational Semantics

$$\text{(OFifo0)} \frac{}{\langle A \text{--}\square\text{--}\rightarrow B, \{A \mapsto u \wedge d\} \uplus \sigma \rangle \Rightarrow \langle A \text{--}d\text{--}\rightarrow B, \{A \mapsto u\} \uplus \sigma \rangle}$$

$$\text{(OFifo1)} \frac{}{\langle A \text{--}d\text{--}\rightarrow B, \{B \mapsto u\} \uplus \sigma \rangle \Rightarrow \langle A \text{--}\square\text{--}\rightarrow B, \{B \mapsto d \wedge u\} \uplus \sigma \rangle}$$



Operational Semantics

$$\text{(OFifo)} \frac{}{\langle A \text{--}\square\text{--}\rightarrow B, \{A \mapsto u \frown d\} \uplus \sigma \rangle \Rightarrow \langle A \text{--}d\text{--}\rightarrow B, \{A \mapsto u\} \uplus \sigma \rangle}$$

$$\text{(OFifo)} \frac{}{\langle A \text{--}d\text{--}\rightarrow B, \{B \mapsto u\} \uplus \sigma \rangle \Rightarrow \langle A \text{--}\square\text{--}\rightarrow B, \{B \mapsto d \frown u\} \uplus \sigma \rangle}$$

$$\text{(Syn)} \frac{}{\langle A \mapsto B, \{A \mapsto u \frown d, B \mapsto v\} \uplus \sigma \rangle \Rightarrow \langle A \mapsto B, \{A \mapsto u, B \mapsto d \frown v\} \uplus \sigma \rangle}$$

Operational Semantics

$$\text{(OFifoo)} \frac{}{\langle A \neg \square \rightarrow B, \{A \mapsto u \wedge d\} \uplus \sigma \rangle \Rightarrow \langle A \neg d \rightarrow B, \{A \mapsto u\} \uplus \sigma \rangle}$$

$$\text{(OFifo1)} \frac{}{\langle A \neg d \rightarrow B, \{B \mapsto u\} \uplus \sigma \rangle \Rightarrow \langle A \neg \square \rightarrow B, \{B \mapsto d \wedge u\} \uplus \sigma \rangle}$$

$$\text{(Syn)} \frac{}{\langle A \mapsto B, \{A \mapsto u \wedge d, B \mapsto v\} \uplus \sigma \rangle \Rightarrow \langle A \mapsto B, \{A \mapsto u, B \mapsto d \wedge v\} \uplus \sigma \rangle}$$

$$\text{(Synd)} \frac{}{\langle (A, B) \succ \emptyset, \{A \mapsto u \wedge d, B \mapsto v \wedge d'\} \uplus \sigma \rangle \Rightarrow \langle (A, B) \succ \emptyset, \{A \mapsto u, B \mapsto v\} \uplus \sigma \rangle}$$

Operational Semantics

$$\text{(IFifo)} \frac{}{\langle A \text{ --}[u] \rightarrow B, \{A \mapsto v \frown d\} \uplus \sigma \rangle \Rightarrow \langle A \text{ --}[d \frown u] \rightarrow B, \{A \mapsto v\} \uplus \sigma \rangle}$$

$$\text{(IFifoI)} \frac{}{\langle A \text{ --}[u \frown d] \rightarrow B, \{B \mapsto v\} \uplus \sigma \rangle \Rightarrow \langle A \text{ --}[u] \rightarrow B, \{B \mapsto d \frown v\} \uplus \sigma \rangle}$$

Operational Semantics

$$\text{(IFifo)} \frac{}{\langle A \text{ --}[u]\text{--} B, \{A \mapsto v \frown d\} \uplus \sigma \rangle \Rightarrow \langle A \text{ --}[d \frown u]\text{--} B, \{A \mapsto v\} \uplus \sigma \rangle}$$

$$\text{(IFifoI)} \frac{}{\langle A \text{ --}[u \frown d]\text{--} B, \{B \mapsto v\} \uplus \sigma \rangle \Rightarrow \langle A \text{ --}[u]\text{--} B, \{B \mapsto d \frown v\} \uplus \sigma \rangle}$$

$$\text{(Fr)} \frac{}{\langle A \text{ --}\langle B, C \rangle\text{--}, \{A \mapsto u \frown d, B \mapsto v, C \mapsto w\} \uplus \sigma \rangle \Rightarrow \langle A \text{ --}\langle B, C \rangle\text{--}, \{A \mapsto u, B \mapsto d \frown v, C \mapsto d \frown w\} \uplus \sigma \rangle}$$

Operational Semantics

$$\text{(IFifoo)} \frac{}{\langle A \text{--[}u\text{]}\rightarrow B, \{A \mapsto v \wedge d\} \uplus \sigma \rangle \Rightarrow \langle A \text{--[}d \wedge u\text{]}\rightarrow B, \{A \mapsto v\} \uplus \sigma \rangle}$$

$$\text{(IFifoI)} \frac{}{\langle A \text{--[}u \wedge d\text{]}\rightarrow B, \{B \mapsto v\} \uplus \sigma \rangle \Rightarrow \langle A \text{--[}u\text{]}\rightarrow B, \{B \mapsto d \wedge v\} \uplus \sigma \rangle}$$

$$\text{(Fr)} \frac{}{\langle A \preceq (B, C), \{A \mapsto u \wedge d, B \mapsto v, C \mapsto w\} \uplus \sigma \rangle \Rightarrow \langle A \preceq (B, C), \{A \mapsto u, B \mapsto d \wedge v, C \mapsto d \wedge w\} \uplus \sigma \rangle}$$

$$\text{(Mro)} \frac{}{\langle (A, B) \succcurlyeq C, \{A \mapsto u \wedge d, B \mapsto v, C \mapsto w\} \uplus \sigma \rangle \Rightarrow \langle (A, B) \succcurlyeq C, \{A \mapsto u, B \mapsto v, C \mapsto d \wedge w\} \uplus \sigma \rangle}$$

Operational Semantics

$$\text{(Join)} \frac{\begin{array}{l} \langle sys_0, \sigma \rangle \Rightarrow \langle sys'_0, \sigma' \rangle \\ \langle sys_1, \sigma' \rangle \Rightarrow \langle sys'_1, \sigma'' \rangle \\ sys_0 \cap sys_1 = \emptyset \quad \forall_{x \in hid(sys \cup)} \sigma''(x) = \square \end{array}}{\langle sys_0 \cup sys_1, \sigma \rangle \Rightarrow \langle sys'_0 \cup sys'_1, \sigma'' \rangle}$$

Operational Semantics

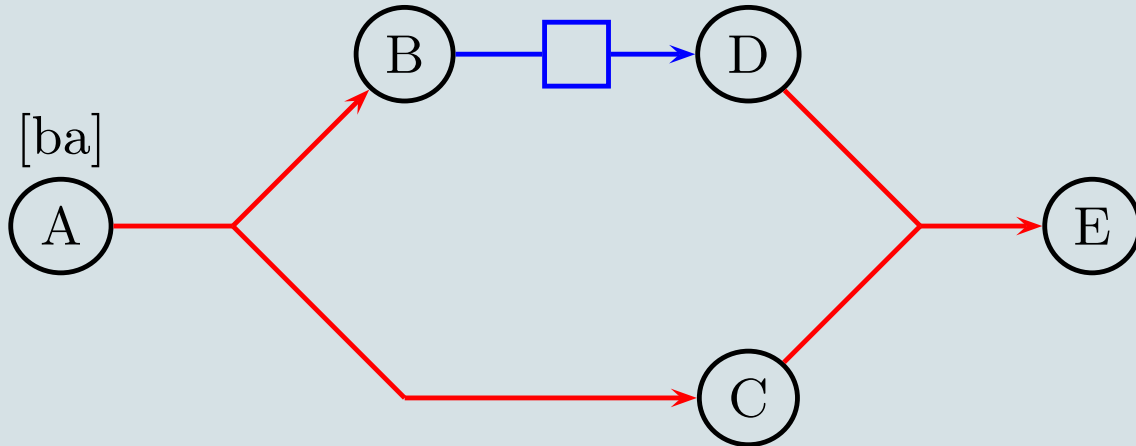
$$\begin{array}{c}
 \langle sys_0, \sigma \rangle \Rightarrow \langle sys'_0, \sigma' \rangle \\
 \langle sys_1, \sigma' \rangle \Rightarrow \langle sys'_1, \sigma'' \rangle \\
 sys_0 \cap sys_1 = \emptyset \quad \forall_{x \in hid(sys \cup sys')} \sigma''(x) = \square \\
 \text{(Join)} \frac{}{\langle sys_0 \cup sys_1, \sigma \rangle \Rightarrow \langle sys'_0 \cup sys'_1, \sigma'' \rangle}
 \end{array}$$

$$\begin{array}{c}
 \langle sys_0, \sigma \rangle \Rightarrow \langle sys'_0, \sigma' \rangle \\
 sys_0 \subseteq sys \quad \forall_{x \in hid(sys)} \sigma'(x) = \square \\
 \text{(Subsys)} \frac{}{\langle sys_0, \sigma \rangle \rightarrow_{\subseteq sys} \langle sys'_0, \sigma' \rangle}
 \end{array}$$

Operational Semantics

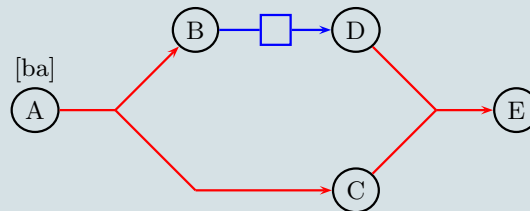
$$\begin{array}{c}
 \langle sys_0, \sigma \rangle \rightarrow_{\subseteq sys_0 \cup sys_1} \langle sys'_0, \sigma' \rangle \\
 sys_0 \cap sys_1 = \emptyset \\
 \forall_{sys_2 \subseteq sys_0 \cup sys_1} sys_1 \subset sys_2 \Rightarrow sys_2 \not\rightarrow_{\subseteq sys_0 \cup sys_1} \\
 \text{(Sys)} \frac{}{\langle sys_0 \cup sys_1, \sigma \rangle \rightsquigarrow \langle sys'_0 \cup sys_1, \sigma' \rangle}
 \end{array}$$

An Example



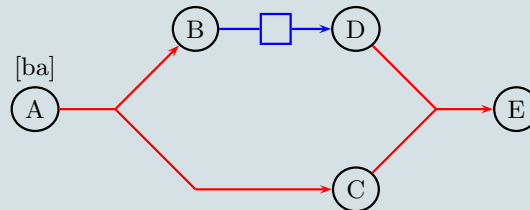
An Example

$$\begin{array}{l} \text{(Fr)} \\ \hline \langle A \preceq (B, C), \\ \{A \mapsto [ba], B \mapsto [], C \mapsto []\} \rangle \Rightarrow \\ \langle A \preceq (B, C), \\ \{A \mapsto [b], B \mapsto [a], C \mapsto [a]\} \rangle \end{array}$$



An Example

$$\begin{array}{c}
 \text{(Fr)} \\
 \hline
 \langle A \preceq (B, C), \\
 \{A \mapsto [ba], B \mapsto [], C \mapsto []\} \rangle \Rightarrow \\
 \langle A \preceq (B, C), \\
 \{A \mapsto [b], B \mapsto [a], C \mapsto [a]\} \rangle
 \end{array}
 \qquad
 \begin{array}{c}
 \text{(Fifoo)} \\
 \hline
 \langle B \xrightarrow{\square} D, \\
 \{B \mapsto [a], D \mapsto []\} \rangle \Rightarrow \\
 \langle B \xrightarrow{a} D, \\
 \{B \mapsto [], D \mapsto []\} \rangle
 \end{array}$$



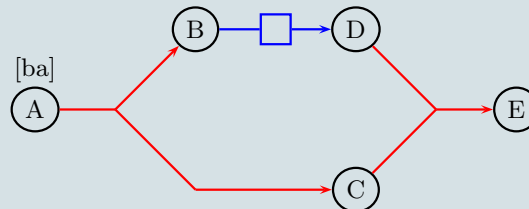
An Example

$$\begin{array}{c}
 \text{(Fr)} \\
 \hline
 \langle A \Leftarrow (B, C), \\
 \{A \mapsto [ba], B \mapsto [], C \mapsto []\} \Rightarrow \\
 \langle A \Leftarrow (B, C), \\
 \{A \mapsto [b], B \mapsto [a], C \mapsto [a]\} \\
 \text{(Fifo)} \\
 \hline
 \langle B \xrightarrow{\square} D, \\
 \{B \mapsto [a], D \mapsto []\} \Rightarrow \\
 \langle B \xrightarrow{a} D, \\
 \{B \mapsto [], D \mapsto []\}
 \end{array}$$

(Join)

$$\begin{array}{c}
 \langle A \Leftarrow (B, C), B \xrightarrow{\square} D, \\
 \{A \mapsto [ba], \dots\} \Rightarrow \Rightarrow \\
 \langle A \Leftarrow (B, C), B \xrightarrow{a} D, \\
 \{A \mapsto [b], \dots\}
 \end{array}$$

(I)

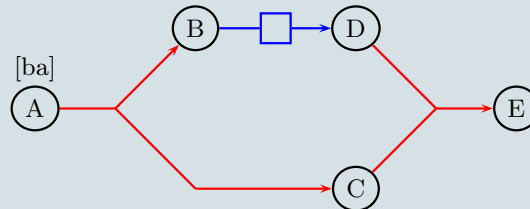


An Example

$$(1) \text{ (Mro)} \frac{\langle (C, D) \succ E, \{C \mapsto [a], D \mapsto [], E \mapsto []\} \rangle \Rightarrow \langle (C, D) \succ E, \{C \mapsto [], D \mapsto [], E \mapsto [a]\} \rangle}{\text{(Join)}}$$

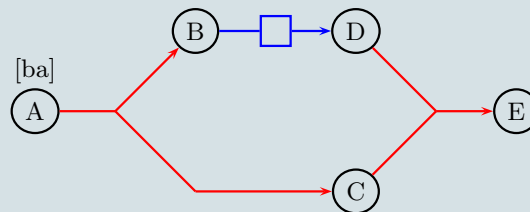
$$\frac{\langle A \leftarrow (B, C), B \dashv \square \rightarrow D, (C, D) \succ E, \{A \mapsto [ba], \dots, E \mapsto []\} \rangle \Rightarrow \langle A \leftarrow (B, C), B \dashv a \rightarrow D, (C, D) \succ E, \{A \mapsto [b], \dots, E \mapsto [a]\} \rangle}{\text{(Join)}}$$

(2)



An Example

$$\begin{array}{c}
 (2) \\
 \hline
 \text{(Sys)} \langle A \preceq (B, C), B \xrightarrow{\square} D, (C, D) \succcurlyeq E, \\
 \{A \mapsto [ba], \dots, E \mapsto []\} \rightsquigarrow \\
 \langle A \preceq (B, C), B \xrightarrow{a} D, (C, D) \succcurlyeq E, \\
 \{A \mapsto [b], \dots, E \mapsto [a]\} \rangle
 \end{array}$$



Bisimulation of Reo Circuits

(Statebased) Simulation Relation R :

for all $(Sys_0, Sys_1) \in R$, $Sink(Sys_0) = Sink(Sys_1)$ and

for all Sys'_0 such that $Sys_0 \rightsquigarrow Sys'_0$ then there exists a system Sys'_1 such that $Sys_1 \rightsquigarrow Sys'_1$ and $(Sys'_0, Sys'_1) \in R$

Two reo circuits are **bisimilar**, denoted by $Sys_0 \leftrightarrow Sys_1$ iff there exists a **symmetric** simulation relation R such that for all node sets X , $(Sys_0 \uparrow X, Sys_1 \uparrow X) \in R$.

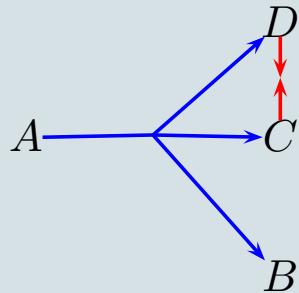
Bisimulation of Reo Circuits

Theorem [Bisimulation](#) on reo circuits is a [congruence](#).

Bisimulation of Reo Circuits: Examples

$A \longrightarrow B$

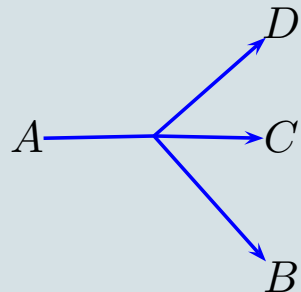
\Leftrightarrow



Bisimulation of Reo Circuits: Examples

$A \longrightarrow B$

$\not\leftrightarrow$



Overview

- ✓ An (Informal) Introduction to Reo
- ✓ Operational Semantics
- From Reo to Maude
- 4. Thoughts on Components and Mobility
- 5. Conclusion and Future Perspective

A Maude Primer

```
fmod VENDING-MACHINE-SIGNATURE is

  sorts Coin Item Marking .
  subsorts Coin Item < Marking .
  op _ _ : Marking Marking ->
          Marking [assoc comm id : null] .
  op null : -> Marking [ctor] .
  op $    : -> Coin [ctor] .
  op q    : -> Coin [ctor] .
  op a    : -> Item [ctor] .
  op c    : -> Item [ctor] .

endfm
```

A Maude Primer

```
mod VENDING-MACHINE is

  protecting VENDING-MACHINE-SIGNATURE .

  var M : Marking .

  rl [add-q]   : M => M q .
  rl [add-$]   : M => M $ .
  rl [buy-c]   : $ => c .
  rl [buy-a]   : q q q => a .
  rl [change]  : $ => q q q q .

endm
```

Implementing SOS in Maude

- Proof derivations as rewrites:
 - rewriting conclusion to premises
 - rewriting premises to conclusion
- Transitions as rewrites

A Maude Primer

```

cr1 [Fifo0] :
    * < (na0, (a ; u)) lFifoE (na1, v) >
      =>
    < (na0, u) lFifoF(a) (na1, v) >

    if

    (a /= emptyEl ).

rl [Fifo1] :
    * < (na0, u) lFifoF(a) (na1, v) >
      =>
    < (na0, u) lFifoE (na1, (v ; a)) > .

```

Reo in Maude

Benefits of the implementation:

1. [Animation](#) (step by step execution) of Reo circuits;
2. Computing [end results](#);
3. Checking for [reachability](#) of a state;
4. [Model checking](#) using LTL formulae.

Reo in Maude

Time for a Demo

Overview

- ✓ An (Informal) Introduction to Reo
- ✓ Operational Semantics
- ✓ From Reo to Maude
- **Thoughts on Components and Mobility**
- 5. Conclusion and Future Perspective

Components: Asymmetry and Anomaly

Asymmetry in our model:

- modeling component writes: implicit by putting initial data on circuit sources
- modeling component reads: not possible! (indeed possible in the original Reo)

Semantics of Lossy Channel

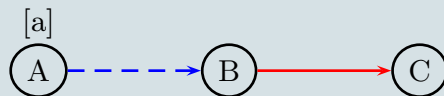
$$\text{(LSyno)} \frac{\langle A \dashrightarrow B, \{A \mapsto u \frown a, b \mapsto v\} \rangle}{\langle A \dashrightarrow B, \{A \mapsto u, b \mapsto a \frown v\} \rangle} \Rightarrow$$

$$\text{(LSynI)} \frac{\langle A \dashrightarrow B, \{A \mapsto u \frown a, b \mapsto v\} \rangle}{\langle A \dashrightarrow B, \{A \mapsto u, b \mapsto v\} \rangle} \Rightarrow$$

Asymmetry and Anomaly!



Asymmetry and Anomaly!



Operational Semantics (reminder)

$$\begin{array}{c} \langle sys_0, \sigma \rangle \rightarrow_{\subseteq sys_0 \cup sys_1} \langle sys'_0, \sigma' \rangle \\ sys_0 \cap sys_1 = \emptyset \\ \forall_{sys_2 \subseteq sys_0 \cup sys_1} sys_1 \subset sys_2 \Rightarrow sys_2 \not\rightarrow_{\subseteq sys_0 \cup sys_1} \\ \text{(Sys)} \frac{}{\langle sys_0 \cup sys_1, \sigma \rangle \rightsquigarrow \langle sys'_0 \cup sys_1, \sigma' \rangle} \end{array}$$

A Component Model for Reo

$$Atomic ::= Skip \mid Delay \mid Read(Name) \mid Write(Name, val)$$
$$Proc ::= Atomic \mid Atomic;Proc \mid Proc\|Proc$$
$$Component ::= \langle NodeSet Proc NodeSet \rangle$$

A Component Model for Reo

$$\frac{\langle ((A, u \frown a) \cup NoS_0) \text{ Read}(A) \text{ Nos}_1 \rangle \Rightarrow}{\langle ((A, u) \cup NoS_0) \text{ Skip } \text{Nos}_1 \rangle}$$

$$\frac{\langle NoS_0 \text{ Write}(A, a) ((A, u) \cup Nos_1) \rangle \Rightarrow}{\langle NoS_0 \text{ Skip } ((A, a \frown u) \cup NoS_1) \rangle}$$

$$\frac{}{\langle NoS_0 \text{ Delay } \text{Nos}_1 \rangle \Rightarrow \langle NoS_0 \text{ Skip } \text{Nos}_1 \rangle}$$

Mobility for Reo

$$MAtomic ::= Skip \mid Delay \mid Move(BConnector \text{ to } BConnector)$$
$$MProc ::= Atomic \mid Atomic; Proc \mid Proc \parallel Proc$$
$$Mobility ::= Sys, MProc$$

Overview

- ✓ An (Informal) Introduction to Reo
- ✓ Operational Semantics
- ✓ From Reo to Maude
- ✓ Thoughts on Components and Mobility
- Conclusion and Future Perspective

Conclusion

1. SOS semantics for (basic) Reo
2. faithful implementation of the semantics in Maude
3. some ideas on components and mobility

Future Work

1. implementing and model checking a case study in the tool
2. writing the semantics of components and mobility and proving desired properties about them
3. implementing these semantics in Maude