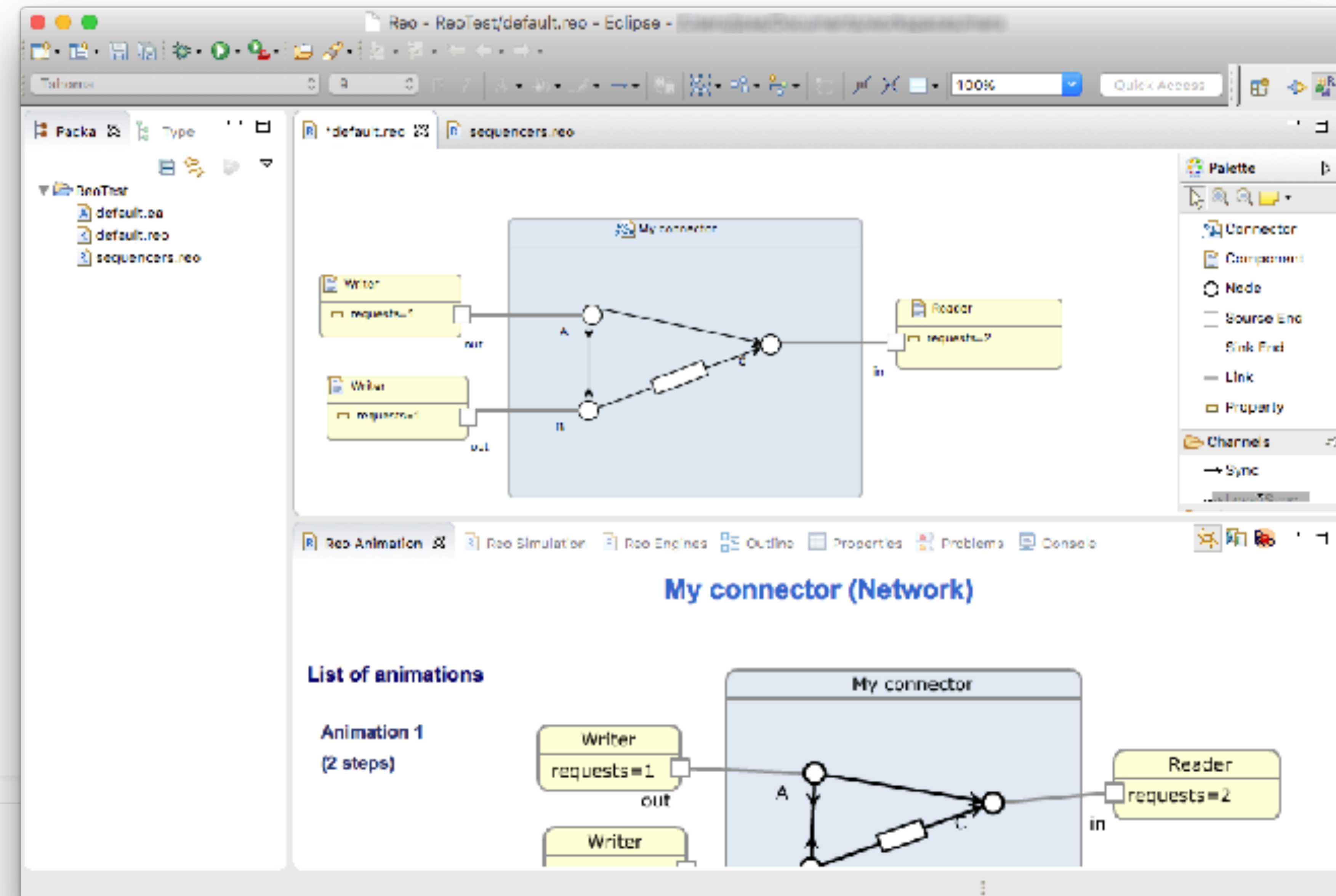
Architectural design: the coordination perspective

José Proença & Luís Soares Barbosa
HASLab - INESC TEC & UM

Arquitectura e Cálculo 2017-18

Reo

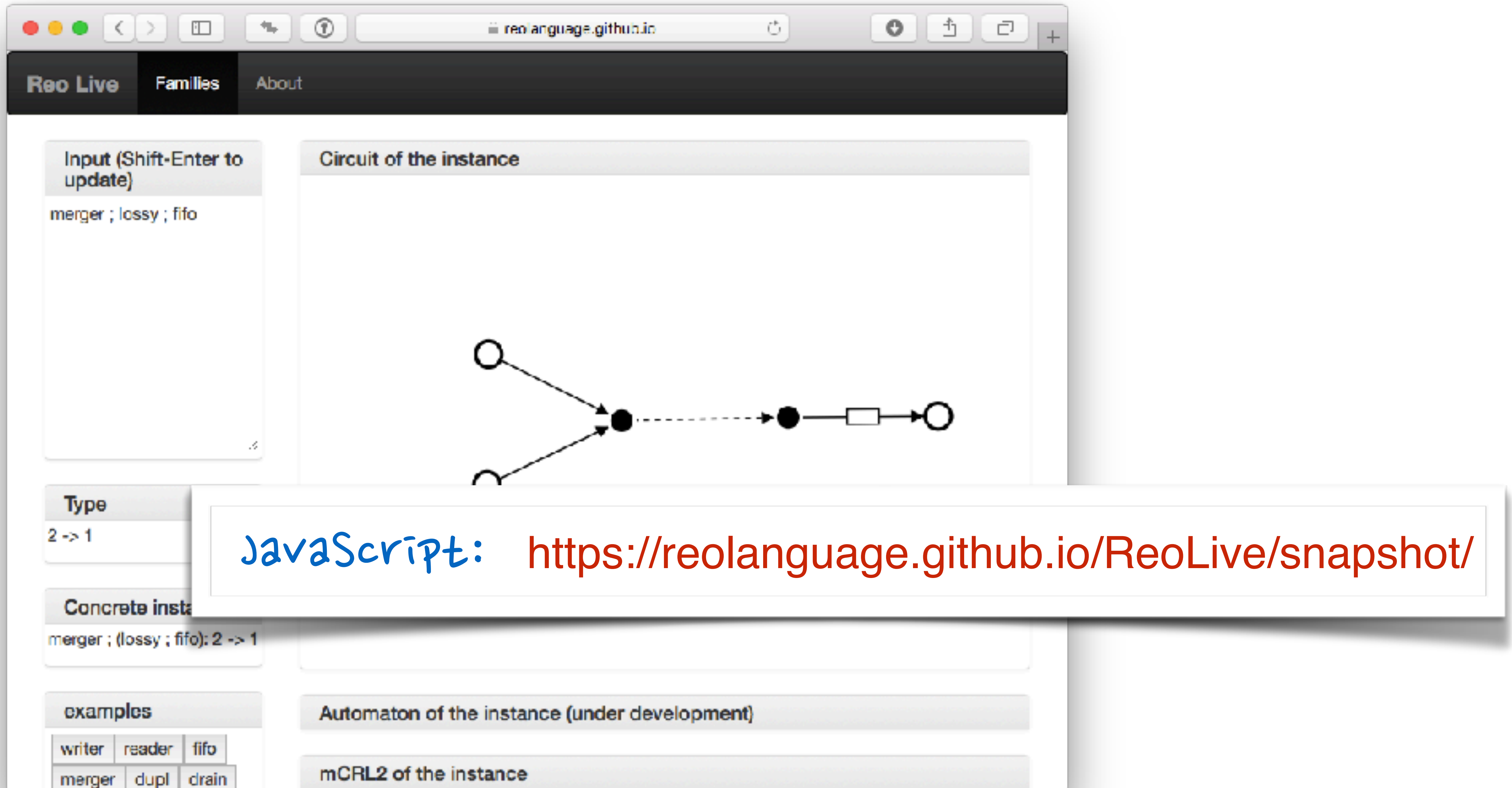
Reo eclipse toolset



get Eclipse

update site: http://reo.project.cwi.nl/update

Reo Live


reolanguage.github.io
Reo Live
Families
About
Input (Shift-Enter to update)
merger ; lossy ; fifo
Circuit of the instance
Type
2 -> 1
Concrete insta
merger ; (lossy ; fifo): 2 -> 1
examples
writer
reader
fifo
merger
dupl
drain
Automaton of the instance (under development)
mCRL2 of the instance
JavaScript:
https://reolanguage.github.io/ReoLive/snapshot/

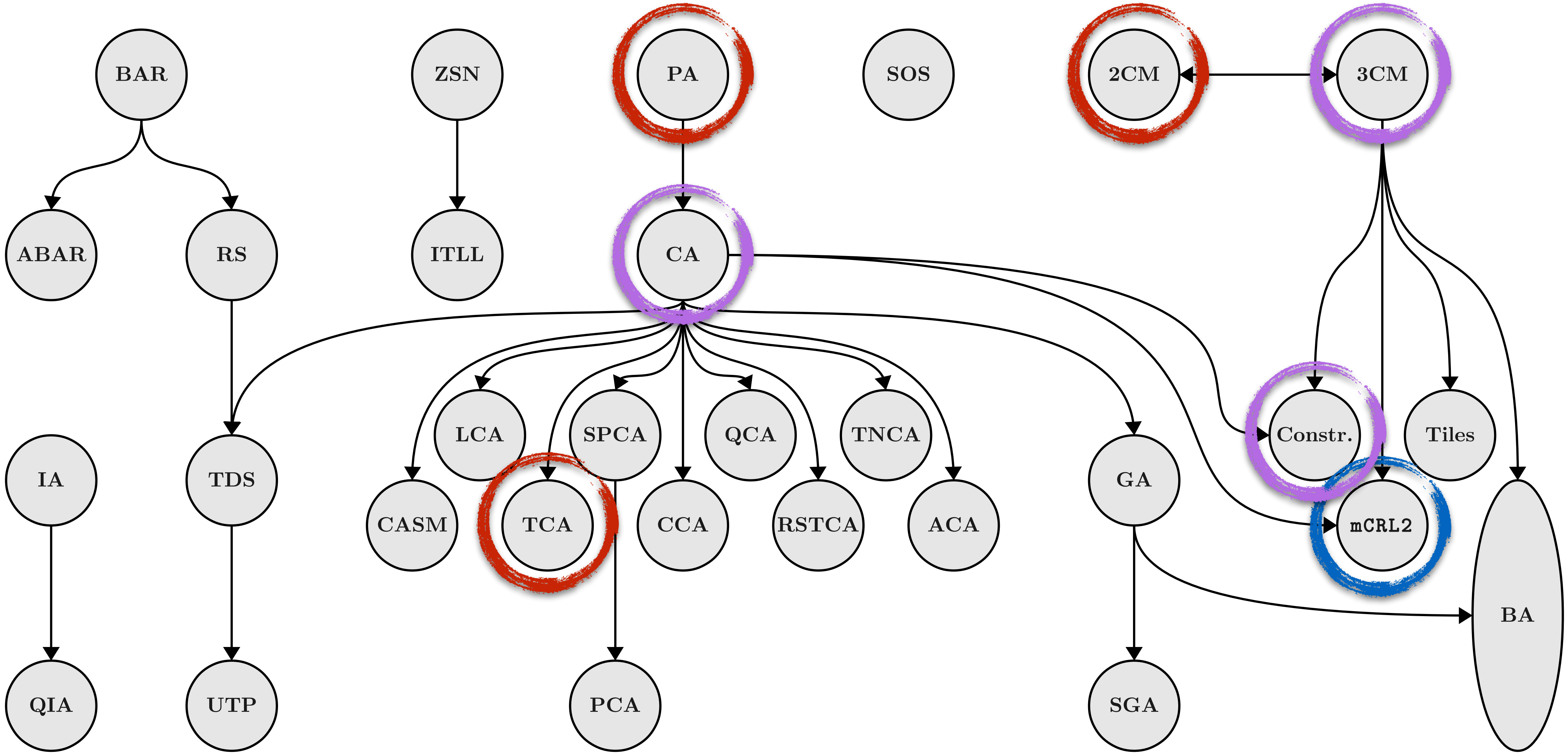
Reo semantics

Jongmans and Arbab 2012

Overview of Thirty Semantic Formalisms for Reo

Reo semantics

- *Coalgebraic models*
  - Timed data streams
  - Record streams
- *Coloring models*
  - **Two colors**
  - **Three colors**
  - Tile models
- *Other models*
  - **Process algebra**
  - **Constraints**
  - Petri nets & intuitionistic logic
  - Unifying theories of programming
  - Structural operational semantics
- *Operational models*
  - **Constraint automata**
  - Variants of constraint automata
    - **Port automata**
    - **Timed**
    - Probabilistic
    - Continuous-time
    - Quantitative
    - Resource-sensitive timed
    - Transactional
  - Context-sensitive automata
    - Büchi automata
    - Reo automata
    - Intentional automata
    - Action constraint automata
    - Behavioral automata
  - Structural operational semantics



2CM : Coloring models with two colors [28, 29, 33]
3CM : Coloring models with three colors [28, 29, 33]
ABAR : Augmented BAR [39, 40]
ACA : Action CA [46]
BA : Behavioral automata [61]
BAR : Büchi automata of records [38, 40]
CA : Constraint automata [10, 17]
CASM : CA with state memory [60]
CCA : Continuous-time CA [18]
Constr. : Propositional constraints [30, 31, 32]
GA : Guarded automata [20, 21]
IA : Intentional automata [33]
ITLL : Intuitionistic temporal linear logic [27]
LCA : Labeled CA [44]
mCRL2 : Process algebra [47, 48, 49]

PA : Port automata [45]
PCA : Probabilistic CA [15]
QCA : Quantitative CA [12, 53]
QIA : Quantitative IA [13]
RS : Record streams [38, 40]
RSTCA : Resource-sensitive timed CA [51]
SGA : Stochastic GA [56, 57]
SOS : Structural operational semantics [58]
SPCA : Simple PCA [15]
TCA : Timed CA [8, 9]
TDS : Timed data streams [4, 5, 14, 62]
Tiles : Tile models [11]
TNCA : Transactional CA [54]
UTP : Unifying theories of programming [55, 52]
ZSN : Zero-safe nets [27]

Outline

Formalism	Synchr.	Data	Time	Context	Partial
Connector Colouring	CC2	-		**CC3**	-
Automata	Port Automata	**Constraint Automata**	**Time CA**	-	-
Constraints	✓	✓	✗	✓	✓

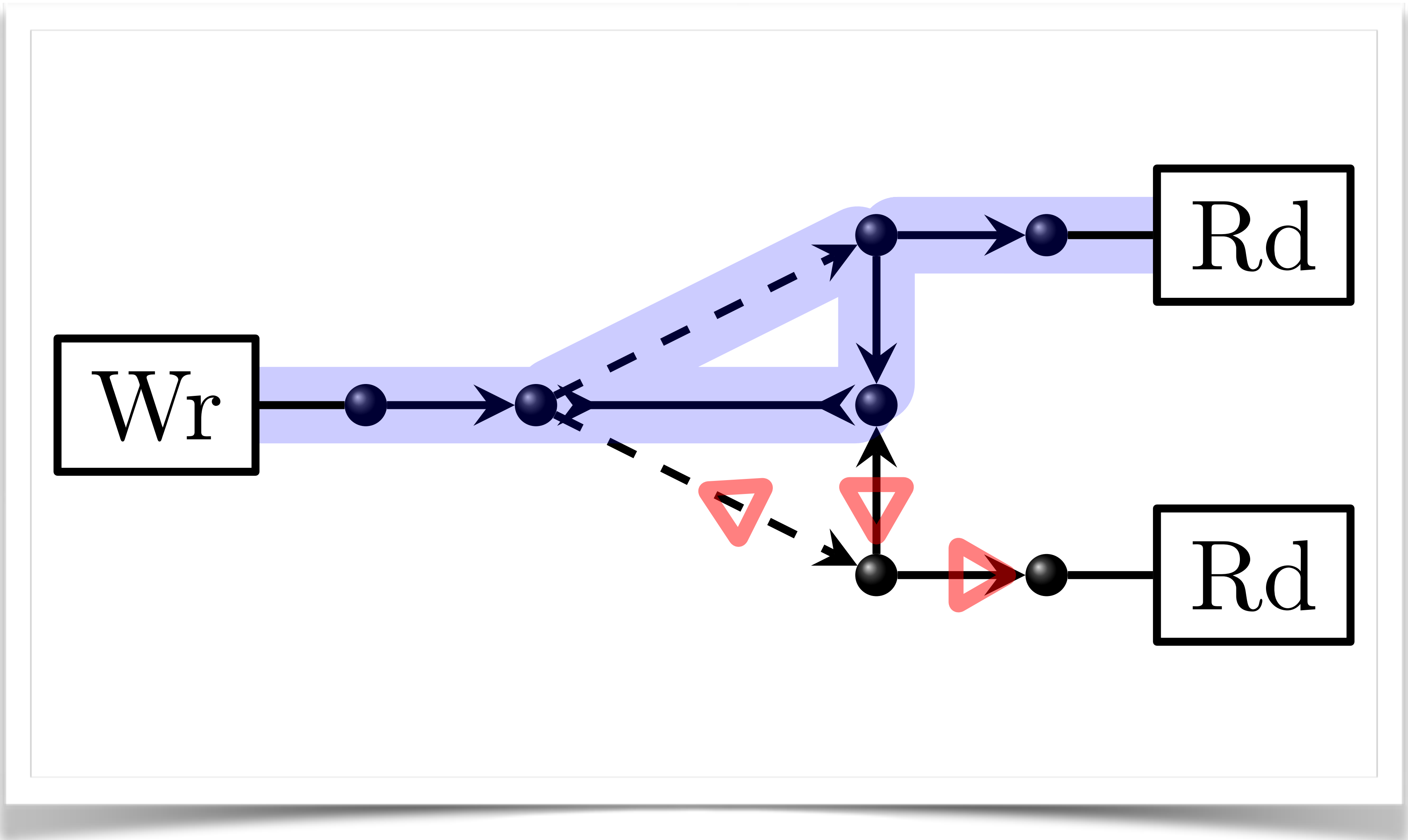
Outline

Intuitive & visual

Formalism	Synchr. 1	Data 2	Time 3	Context	Partial
Connector Colouring	CC2	-		CC3	-
Automata	Port Automata	**Constraint Automata**	**Time CA**	-	-
Constraints	✓	✓	✗	✓	✓

Compilation & verification

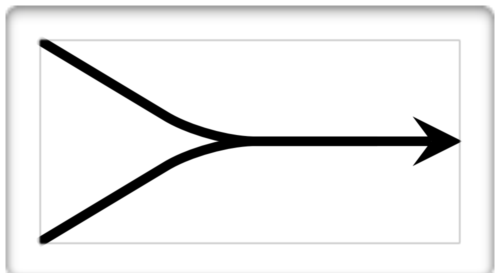
Runtime / scalability



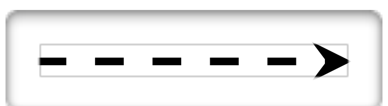
Reo Connector Colouring

Dave Clarke, David Costa, and Farhad Arbab. Connector colouring I: Synchronisation and context dependency

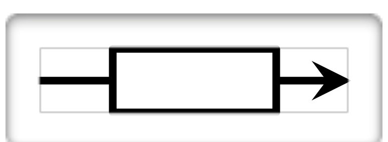
Behaviour?



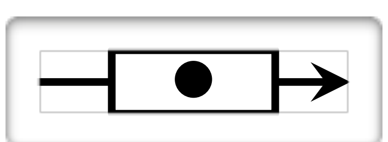
merger: data flows from one of the source ends to the sink end



lossy-sync: either data flows from the source to the sink end, OR it is lost

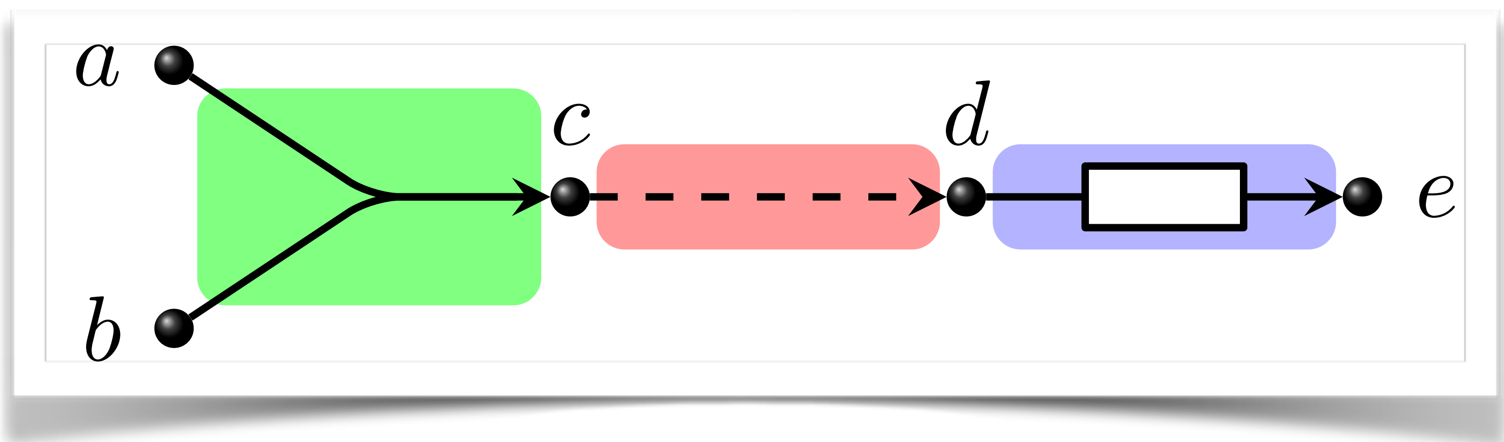


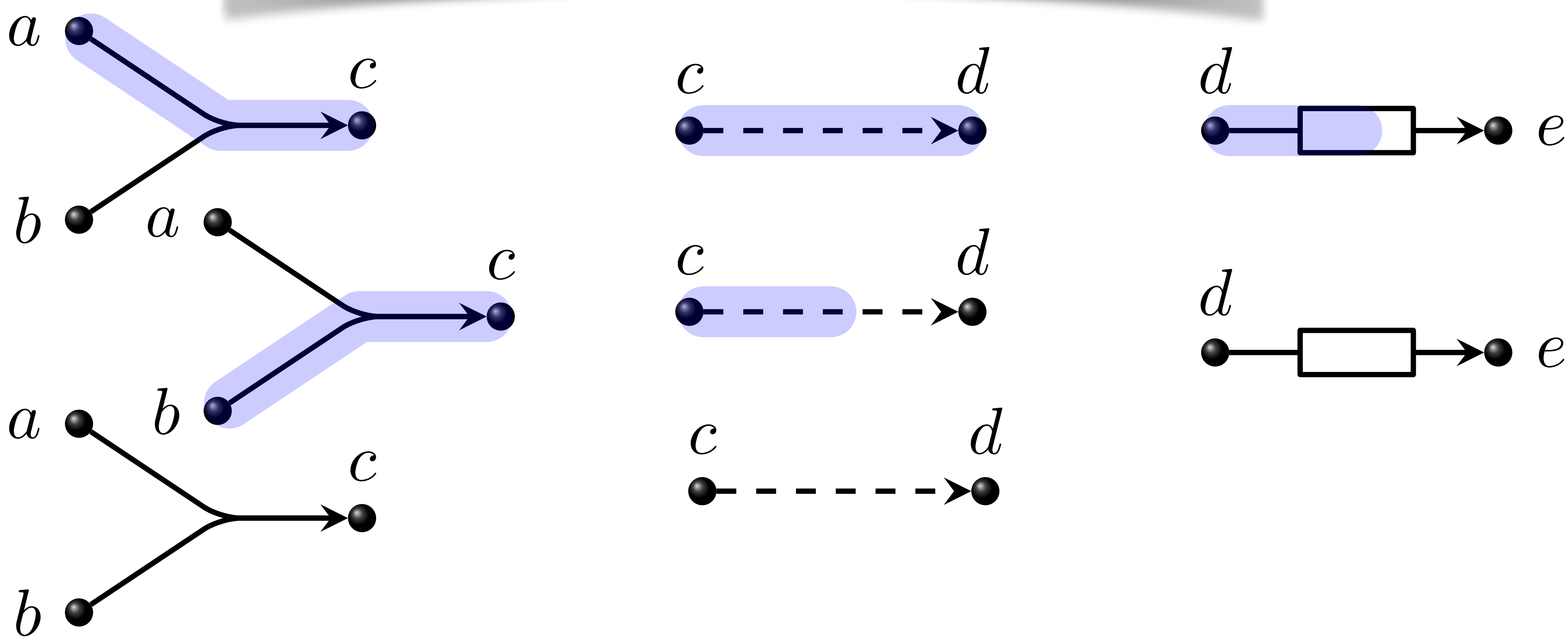
FIFO-1: data flows from the source end to the buffer, becoming a FIFOFull-1



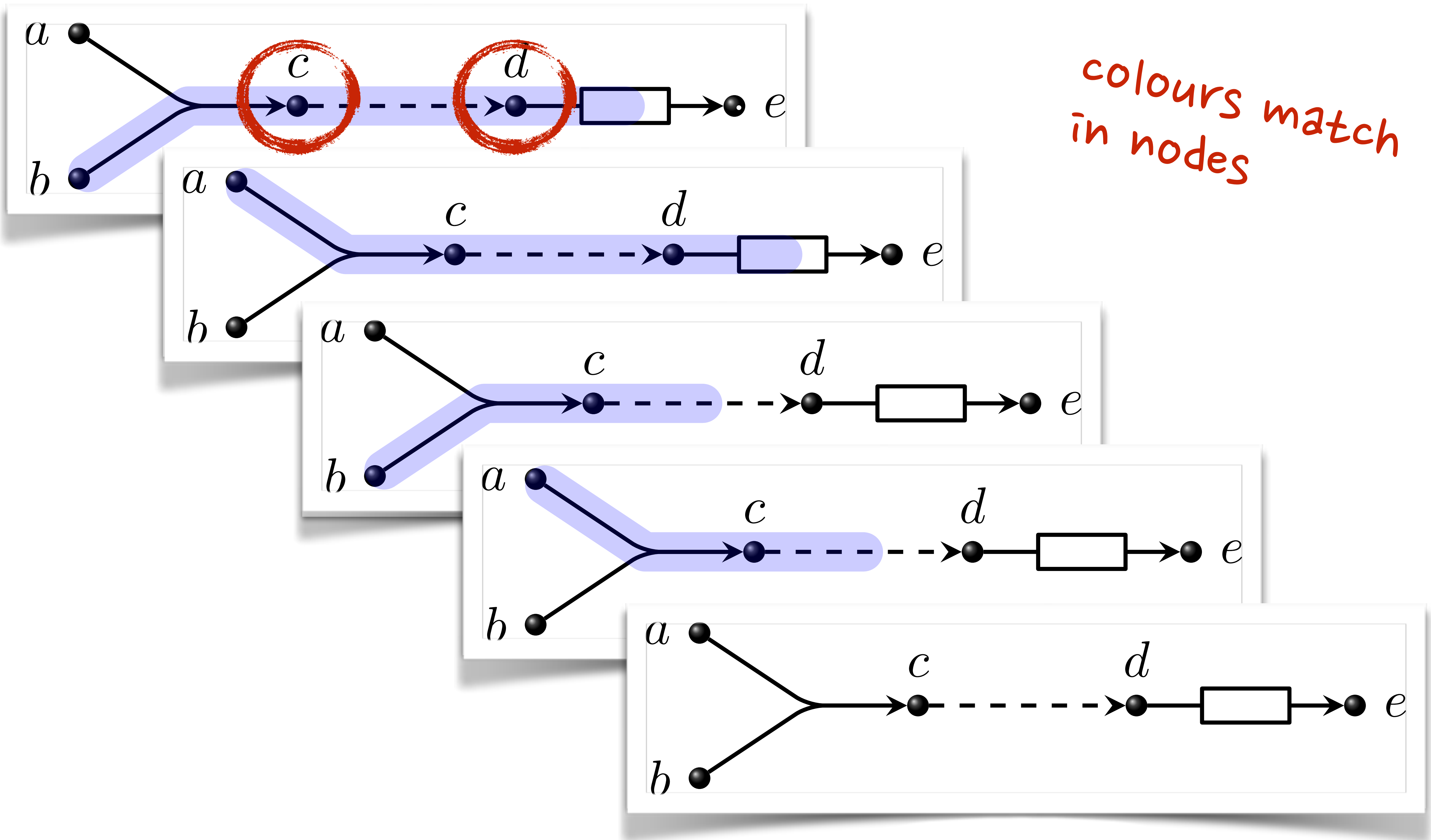
FIFOFull-1: data flows from the buffer to the sink buffer, becoming a FIFO-1

Colourings to describe synchronous dataflow



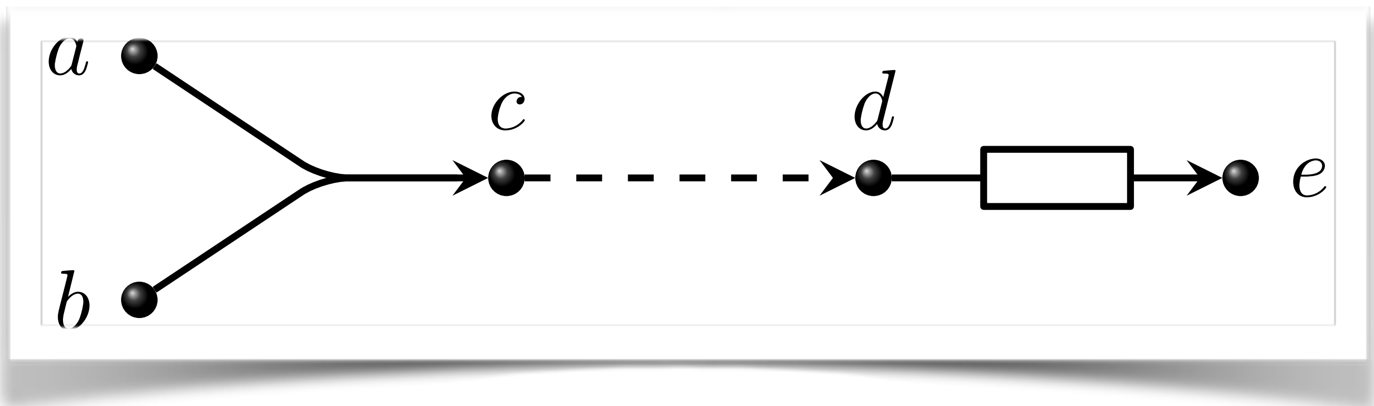


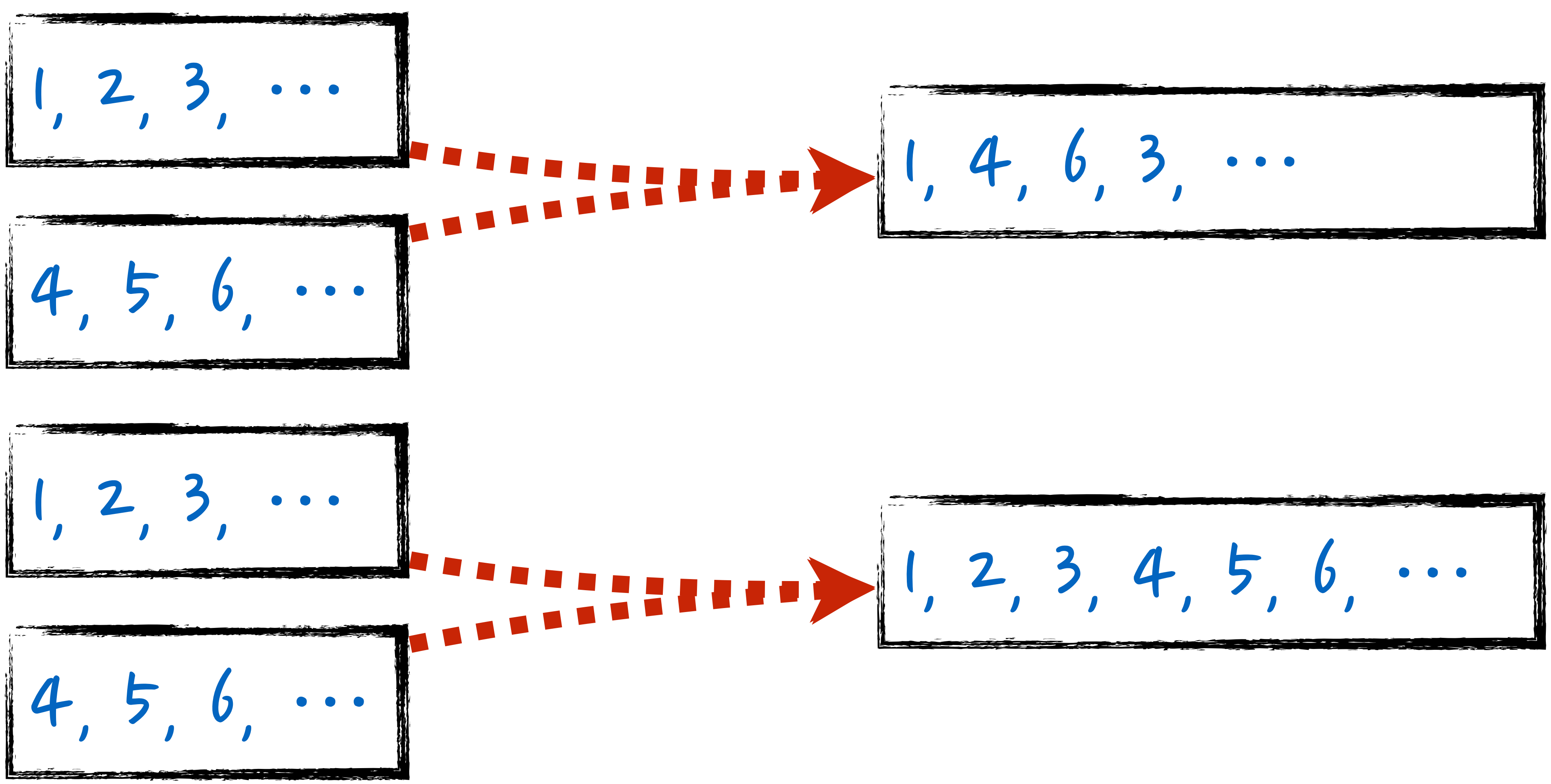
Colouring composition



colours match in nodes

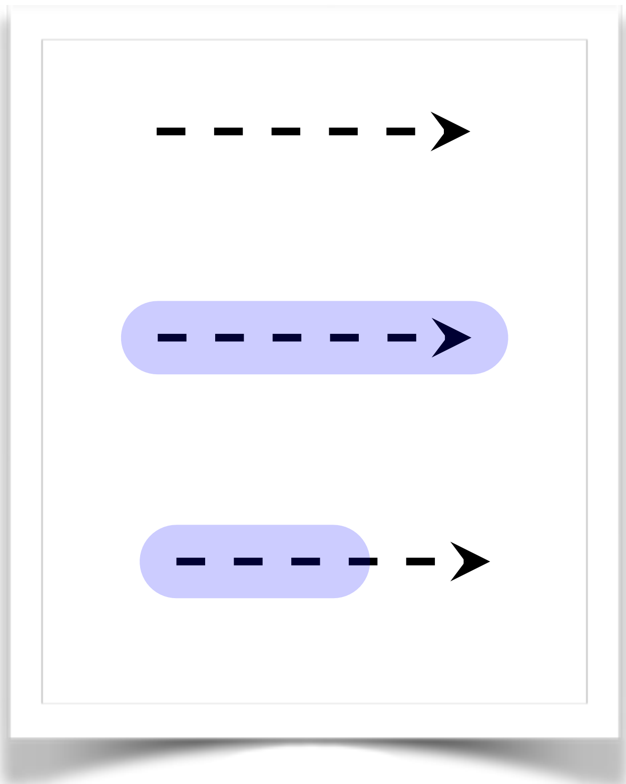
Possible behaviour





Colouring semantics (CC2)

- *Colouring:* End → {Flow, NoFlow}
- *Colouring table:* Set(Colouring)
- *Composition* = matching colours



- More visual (intuitive)
- Used for generating animations

Colouring semantics (CC2)

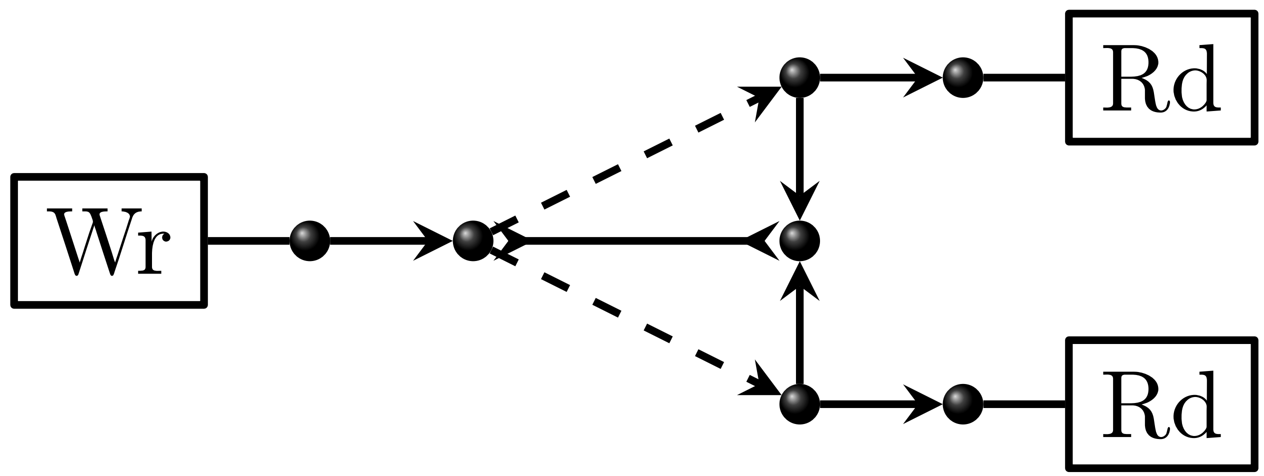
- *Colouring:* End → {Flow, NoFlow}
- *Colouring table:* Set(Colouring)
- *Composition* = matching colours

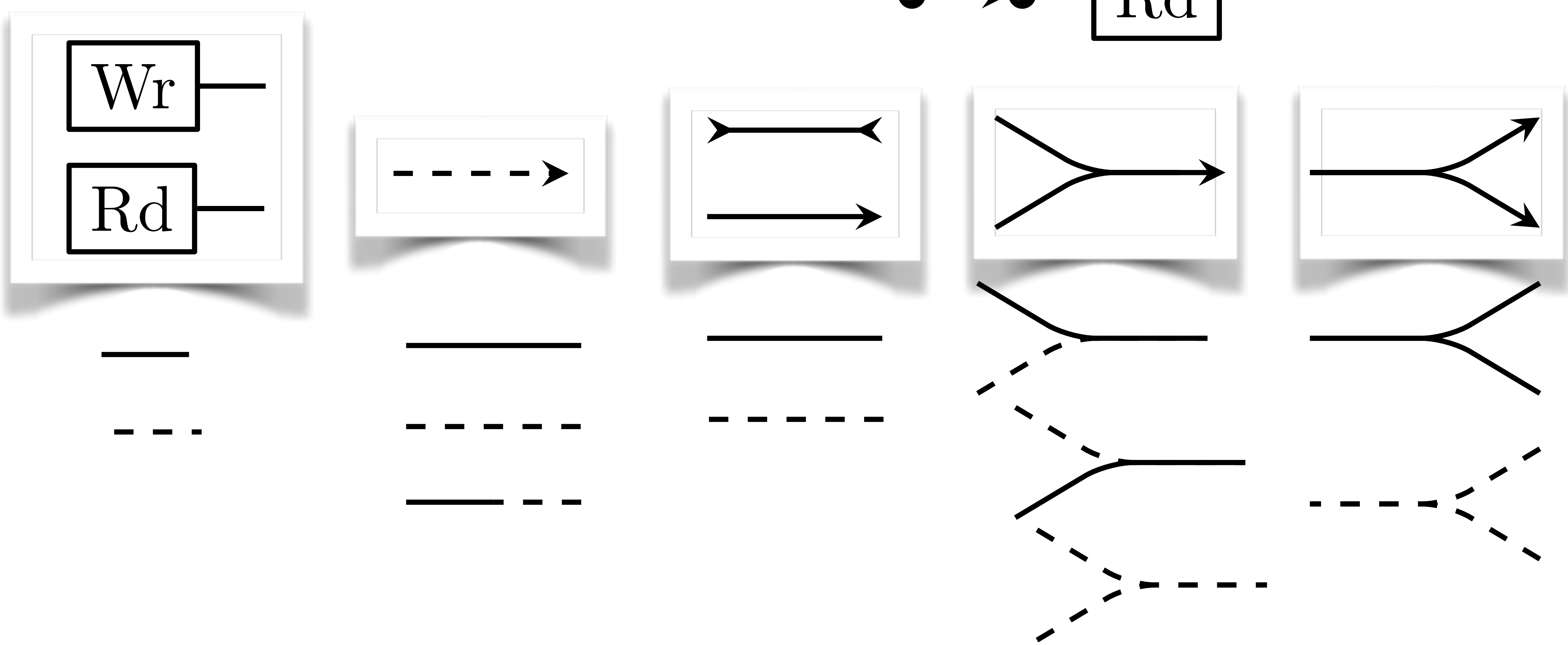
$$CT_1 \bowtie CT_2 \quad = \quad \{cl_1 \bowtie cl_2 \mid cl_1 \in CT_1, cl_2 \in CT_2, cl_1 \frown cl_2\}$$

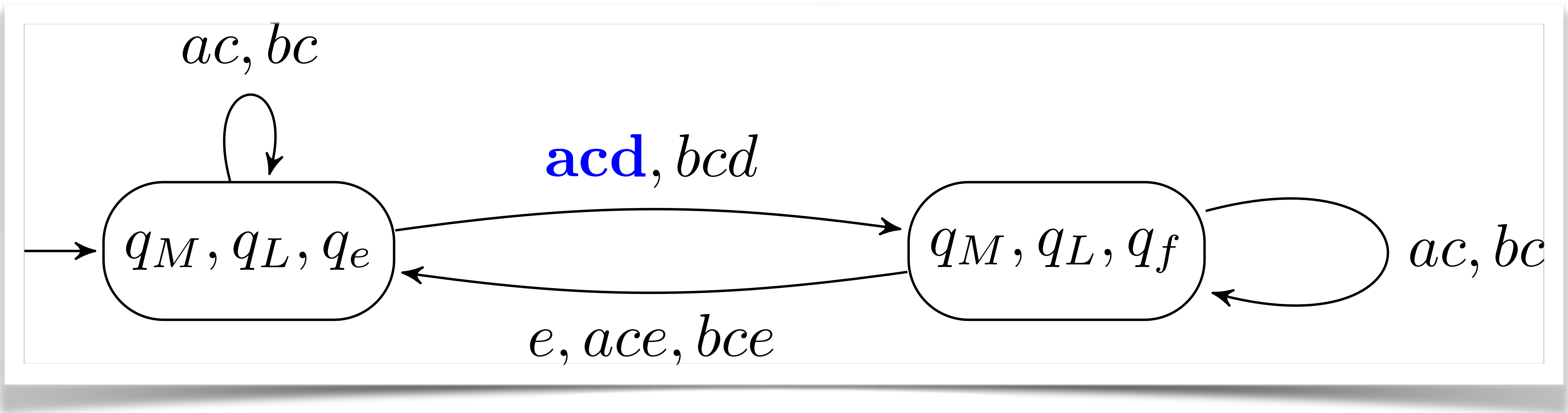
$$cl_1 \frown cl_2 \quad = \quad \forall e \in \mathsf{dom}(cl_1) \cap \mathsf{dom}(cl_2) \cdot cl_1(e) = cl_2(e)$$

$$cl_1 \bowtie cl_2 \quad = \quad cl_1 \cup cl_2$$

Exercise: compose colouring tables







Port Automata

Christian Koehler and Dave Clarke. Decomposing Port Automata. 2009

Connector behaviour (statefull)

- Dataflow behaviour is discrete in time: it can be observed and snapshots taken at a pace fast enough to obtain (at least) a snapshot as often as the configuration of the connector changes
- At each time unit the connector performs an evaluation step: it evaluates its configuration and according to its interaction constraints changes to another (possibly different) configuration
- A connector can fire multiple ports in the same evaluation step

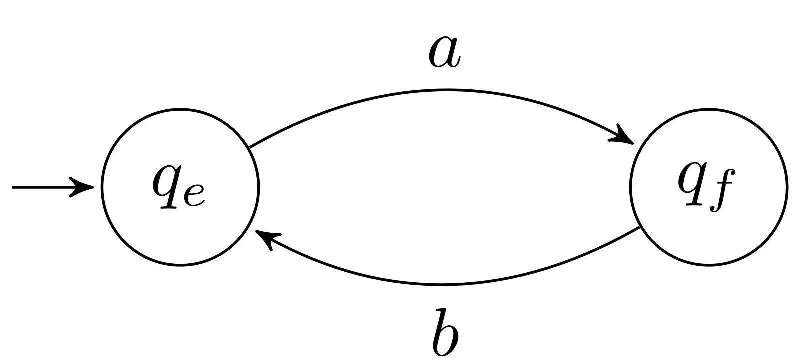
Port Automata

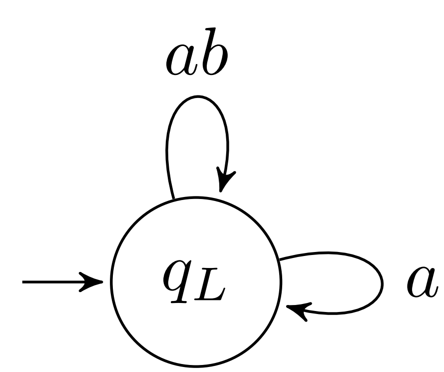
$$\mathcal{A} = (\mathcal{Q}, \mathcal{N}, \rightarrow, \mathcal{Q}_0)$$

$\mathcal{Q}$	set of states
$\mathcal{N}$	a set of ports $\mathcal{N}$
$\rightarrow \;\subseteq \mathcal{Q} \times 2^{\mathcal{N}} \times \mathcal{Q}$	a transition relation
$\mathcal{Q}_0 \subseteq \mathcal{Q}$	a set of initial states

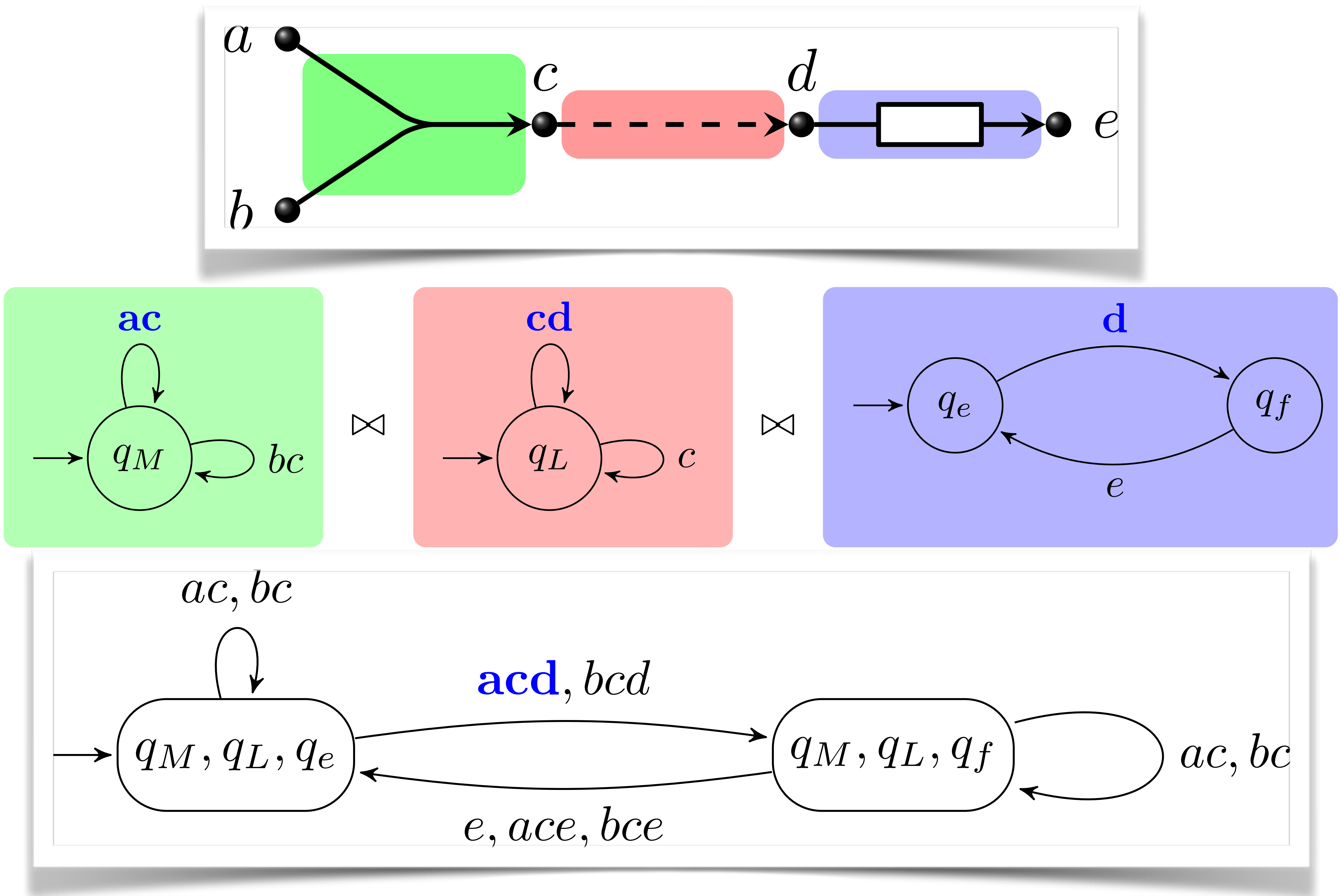
transitions must have a non-empty set of ports!

examples:

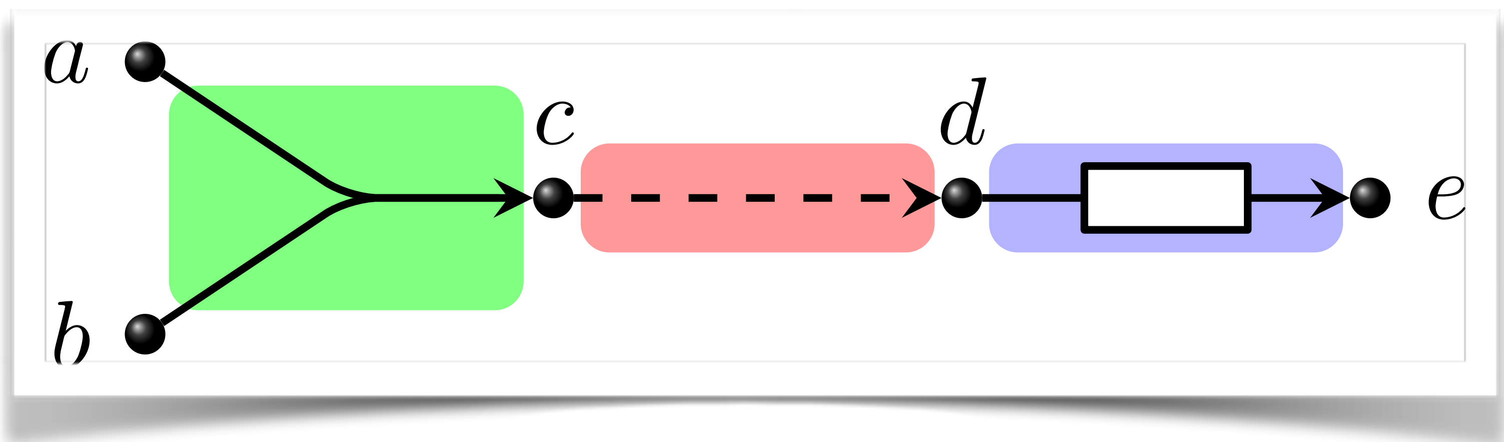


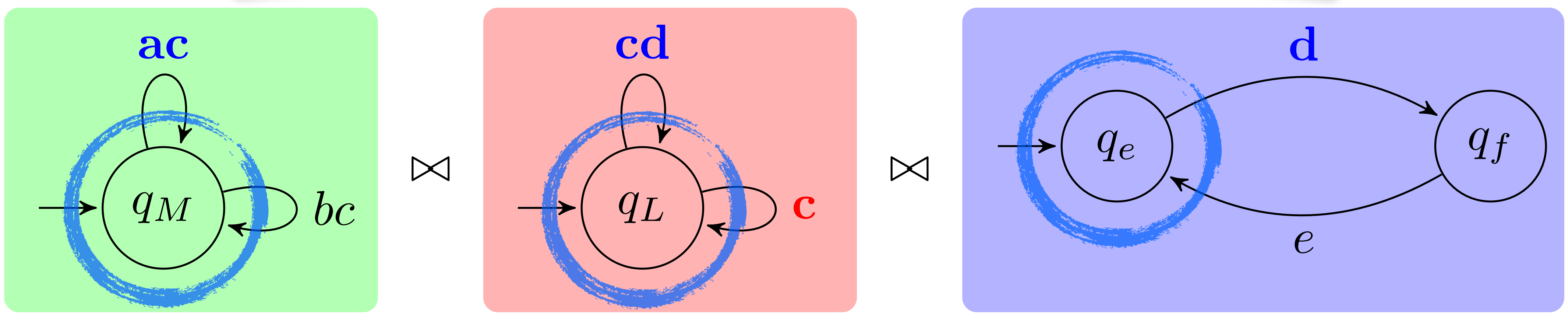


Composing steps



Composing steps

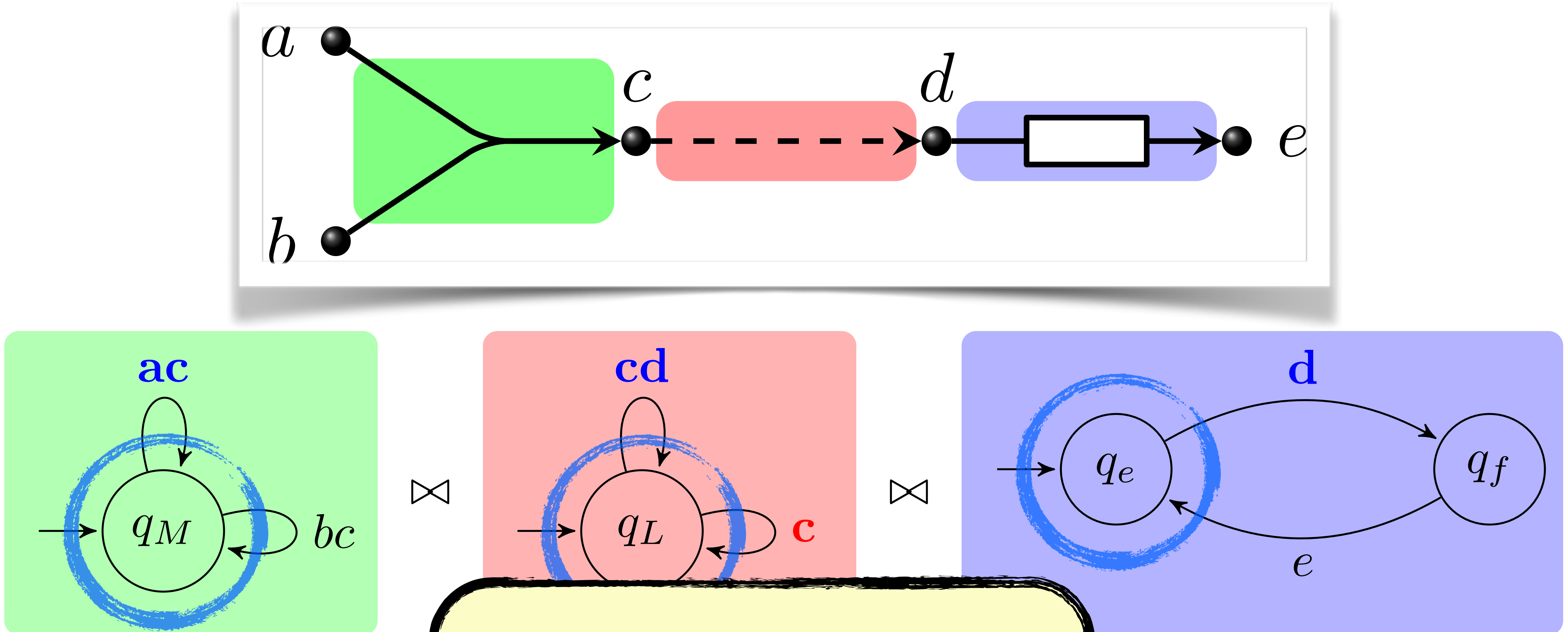




$$\mathbf{ac} \bowtie \mathbf{cd} \bowtie \mathbf{d} \quad = \quad \mathbf{acd}$$

$$\mathbf{ac} \bowtie \mathbf{c} \bowtie \mathbf{d} \quad = \quad \perp$$

Composing steps



merger ; lossy ; fifo

JavaScript: https://reolanguage.github.io/ReoLive/snapshot/

Composition - formally

Definition 2. The product of two port automata $\mathcal{A}_1 = (\mathcal{Q}_1, \mathcal{N}_1, \rightarrow_1, \mathcal{Q}_{0,1})$ and $\mathcal{A}_2 = (\mathcal{Q}_2, \mathcal{N}_2, \rightarrow_2, \mathcal{Q}_{0,2})$ is defined by

$$\mathcal{A}_1 \bowtie \mathcal{A}_2 = (\mathcal{Q}_1 \times \mathcal{Q}_2, \mathcal{N}_1 \cup \mathcal{N}_2, \rightarrow, \mathcal{Q}_{0,1} \times \mathcal{Q}_{0,2})$$

where $\rightarrow$ is defined by the rule

$$\frac{q_1 \xrightarrow{N_1}_1 p_1 \quad q_2 \xrightarrow{N_2}_1 p_2 \quad N_1 \cap \mathcal{N}_2 = N_2 \cap \mathcal{N}_1}{\langle q_1, q_2 \rangle \xrightarrow{N_1 \cup N_2} \langle p_1, p_2 \rangle}$$

and the following and its symmetric rule

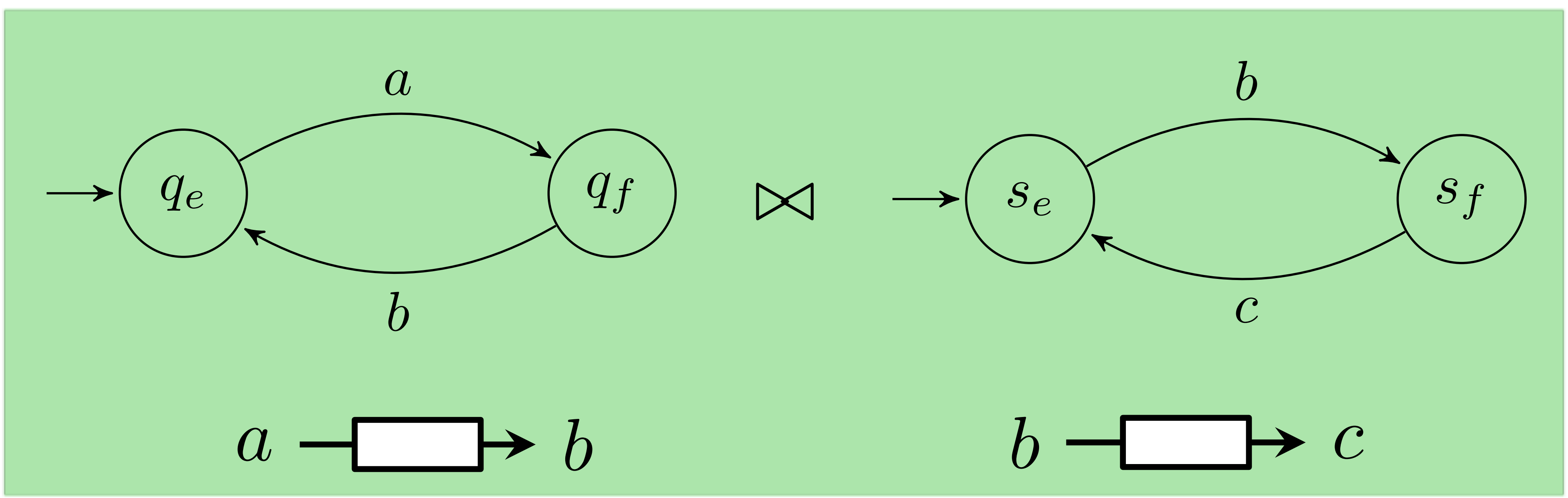
$$\frac{q_1 \xrightarrow{N_1}_1 p_1 \quad N_1 \cap \mathcal{N}_2 = \emptyset}{\langle q_1, q_2 \rangle \xrightarrow{N_1} \langle p_1, q_2 \rangle}$$

Formalize and compose

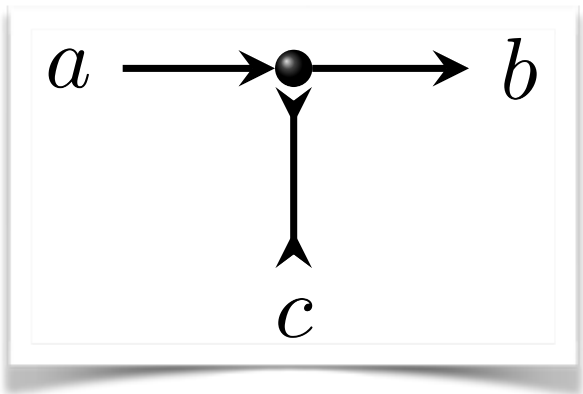
$$\frac{q_1 \xrightarrow{N_1}_1 p_1 \quad q_2 \xrightarrow{N_2}_1 p_2 \quad N_1 \cap \mathcal{N}_2 = N_2 \cap \mathcal{N}_1}{\langle q_1, q_2 \rangle \xrightarrow{N_1 \cup N_2} \langle p_1, p_2 \rangle}$$

$$\mathcal{A} = (\mathcal{Q}, \mathcal{N}, \rightarrow, \mathcal{Q}_0)$$

$$\frac{q_1 \xrightarrow{N_1}_1 p_1 \quad N_1 \cap \mathcal{N}_2 = \emptyset}{\langle q_1, q_2 \rangle \xrightarrow{N_1} \langle p_1, q_2 \rangle}$$

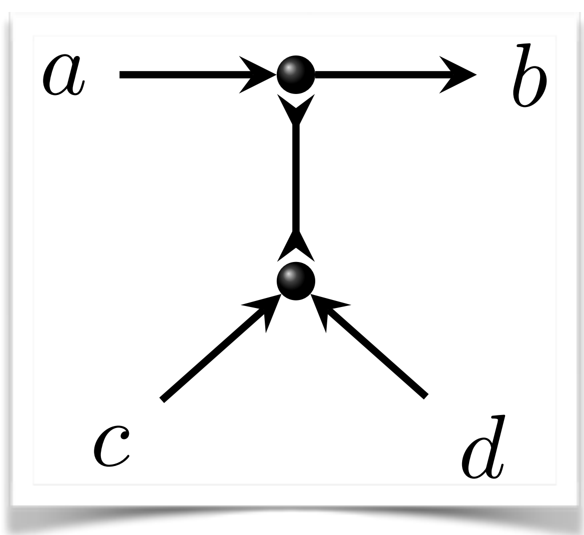


Examples I



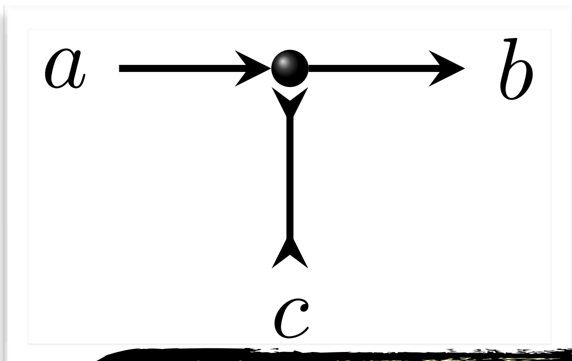
Flow regulator

"c" controls flow from "a" to "b"



data flows from "a" to "b" ONLY if either "c" or "d" have data

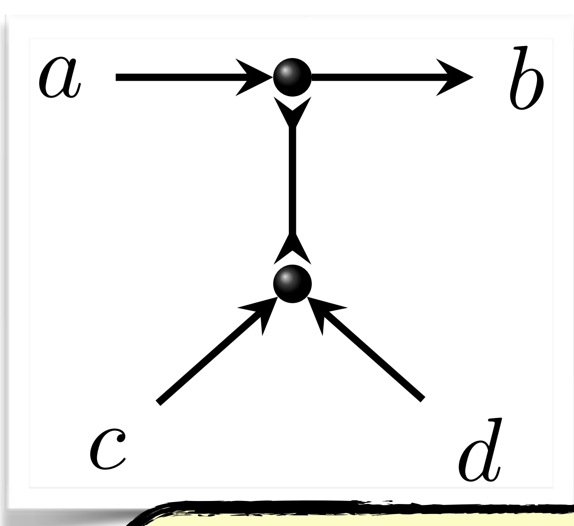
Examples I



dupl*id ; id*drain

Flow regulator

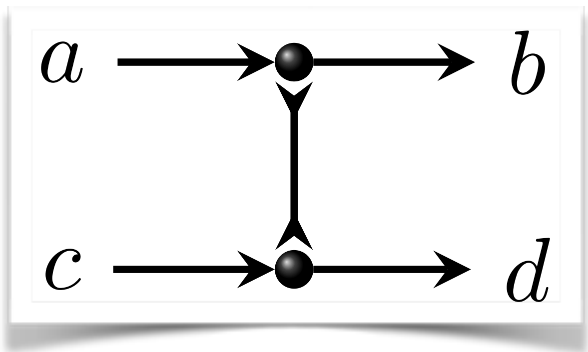
"b" controls flow from "a" to "c"



dupl*merger ; id*drain

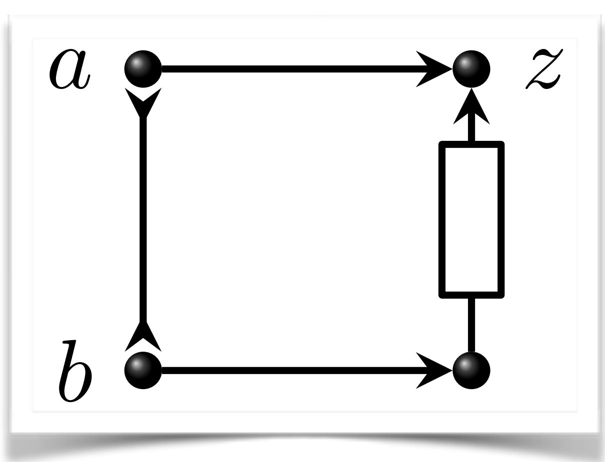
data flows from "a" to "b" ONLY if either "c" or "d" have data

Examples II



Synchronising barrier

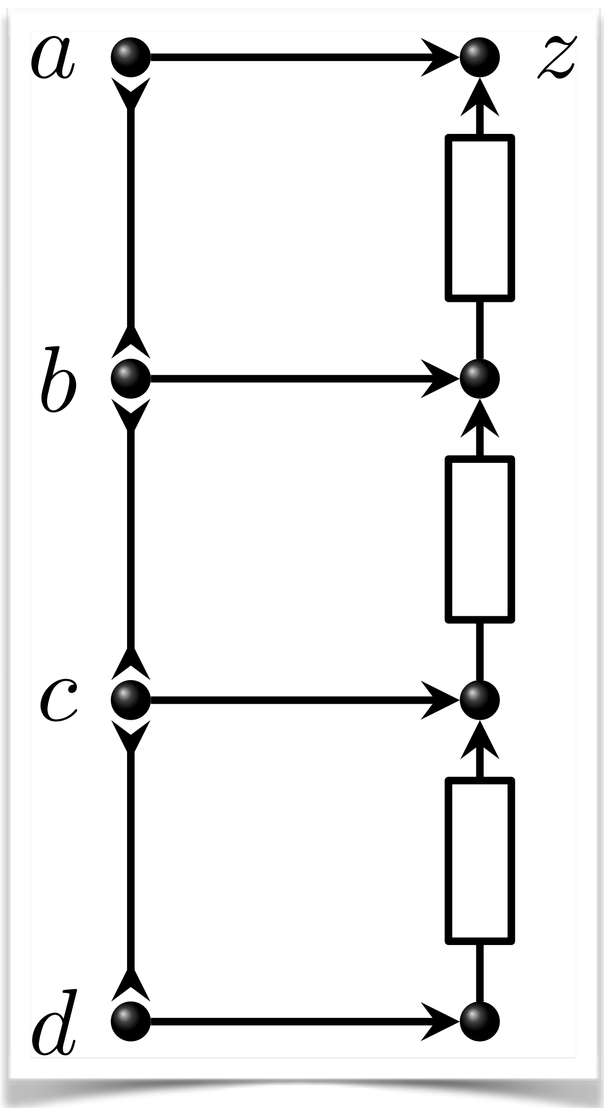
data flows "a" —> "b"
IFF
data flows "c" —> "d"



Alternator

data flows from "a" and from "b" to "z", alternating (+ extra synch constraints)

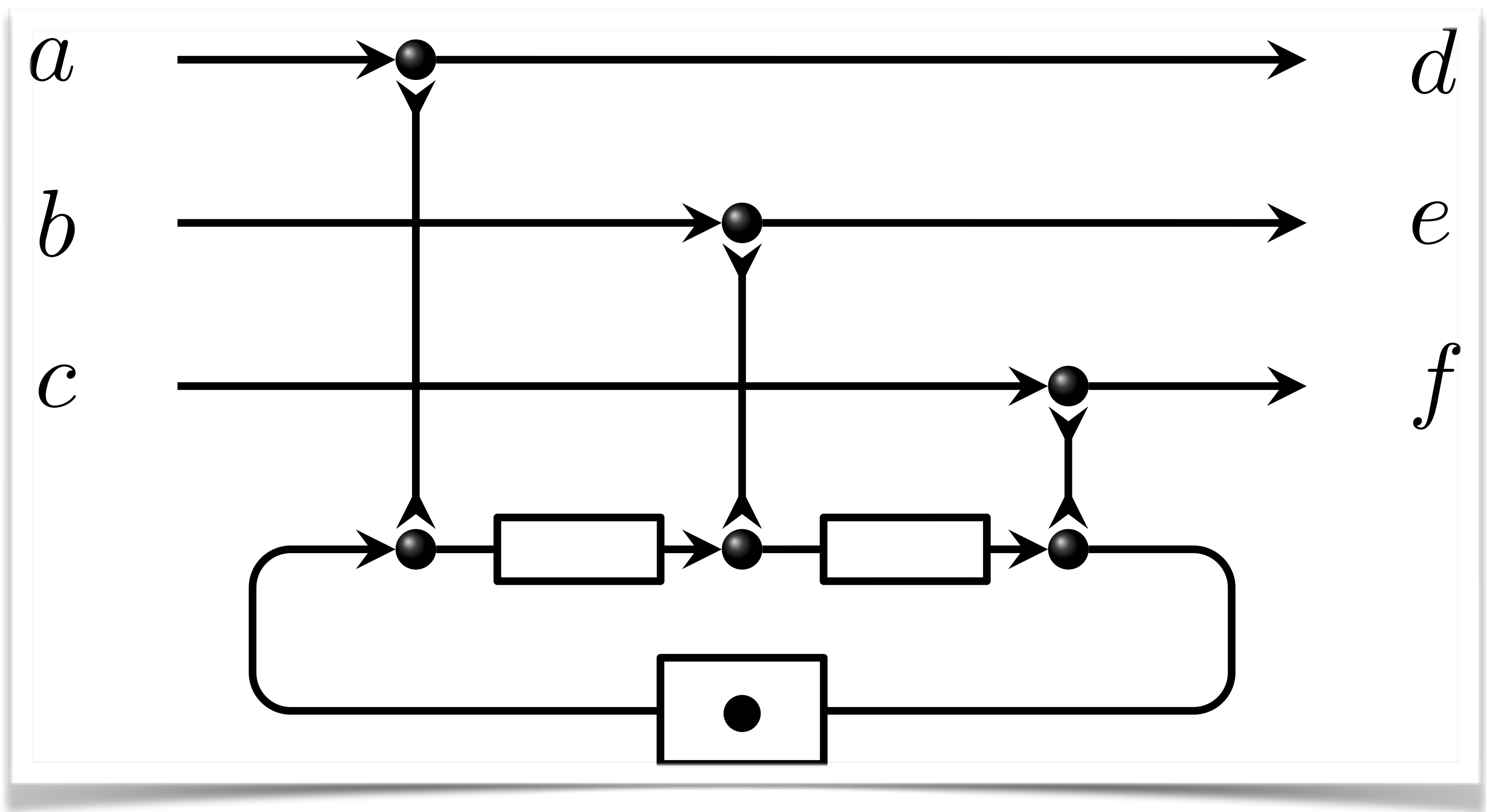
Examples III



N-Alternator

data flows from "a", "b", "c", and "d" to "z", alternating (+ extra synch constraints)

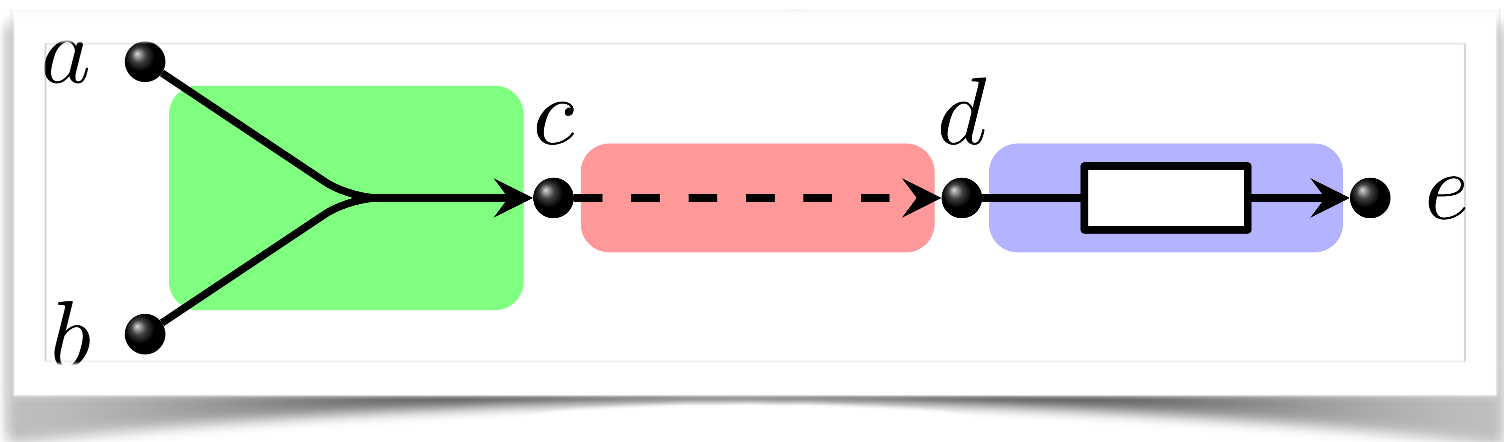
Examples IV

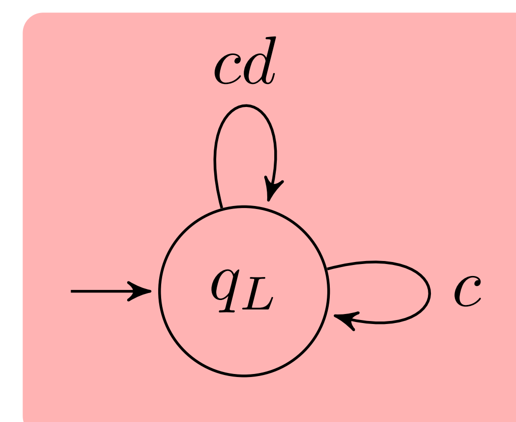


Sequencer

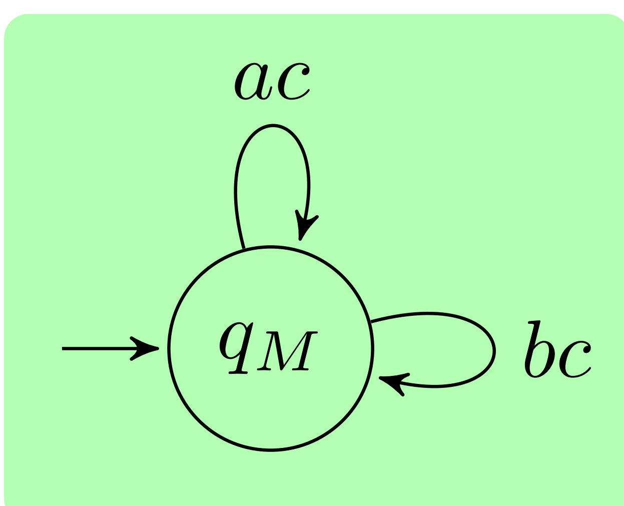
Data flows from "a" to "d", "b" to "e", and "c" to "f" alternating.

Reo in mCRL2



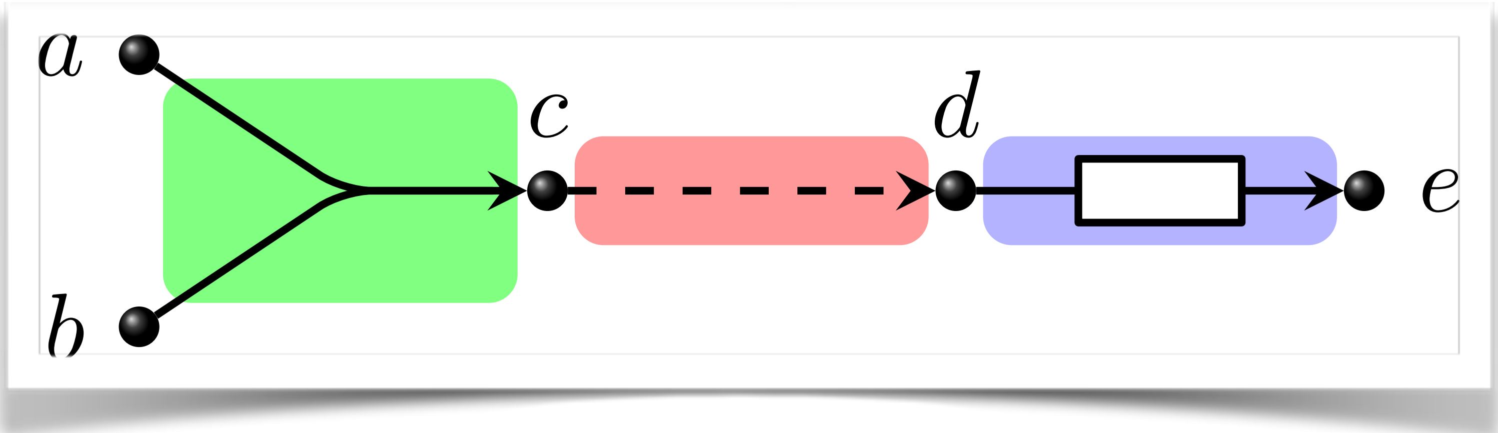


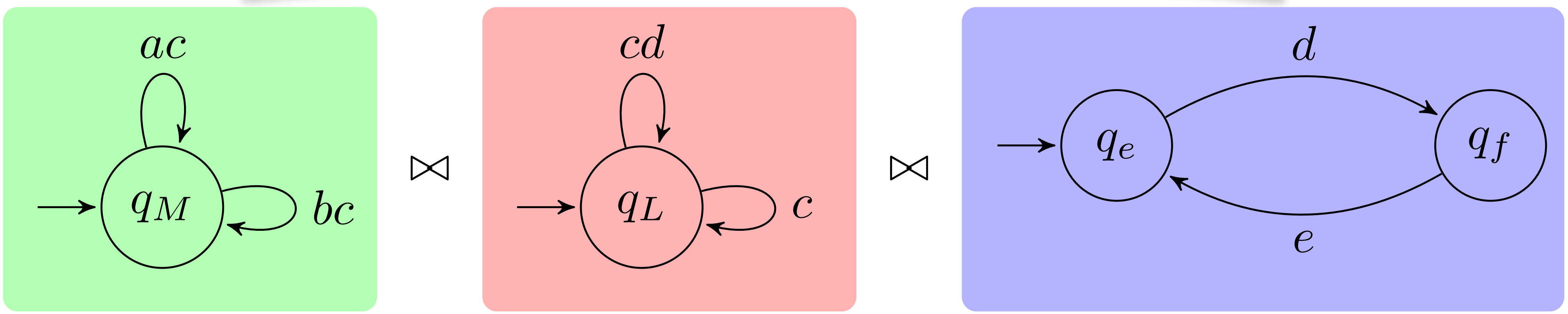
Lossy = ($c|d$ + c).Lossy



Merger = ($a|c$ + $b|c$).Merger

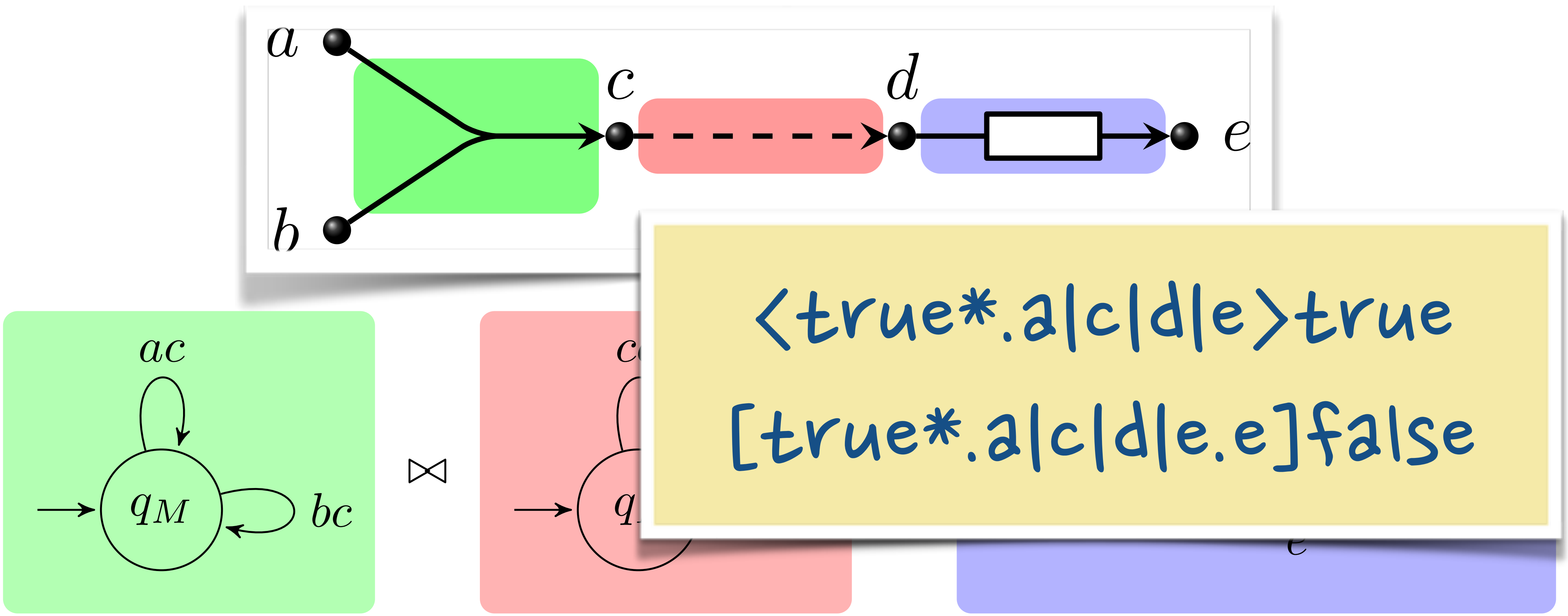
Reo in mCRL2





Conn = hide($\{c,d\}$,
block($\{c_1,c_2,d_1,d_2\}$,
comm($\{c_1|c_2 \to c,\ d_1|d_2 \to \mathbf{d}\}$,
Merger || Lossy || FIFO1)))

Reo in mCRL2

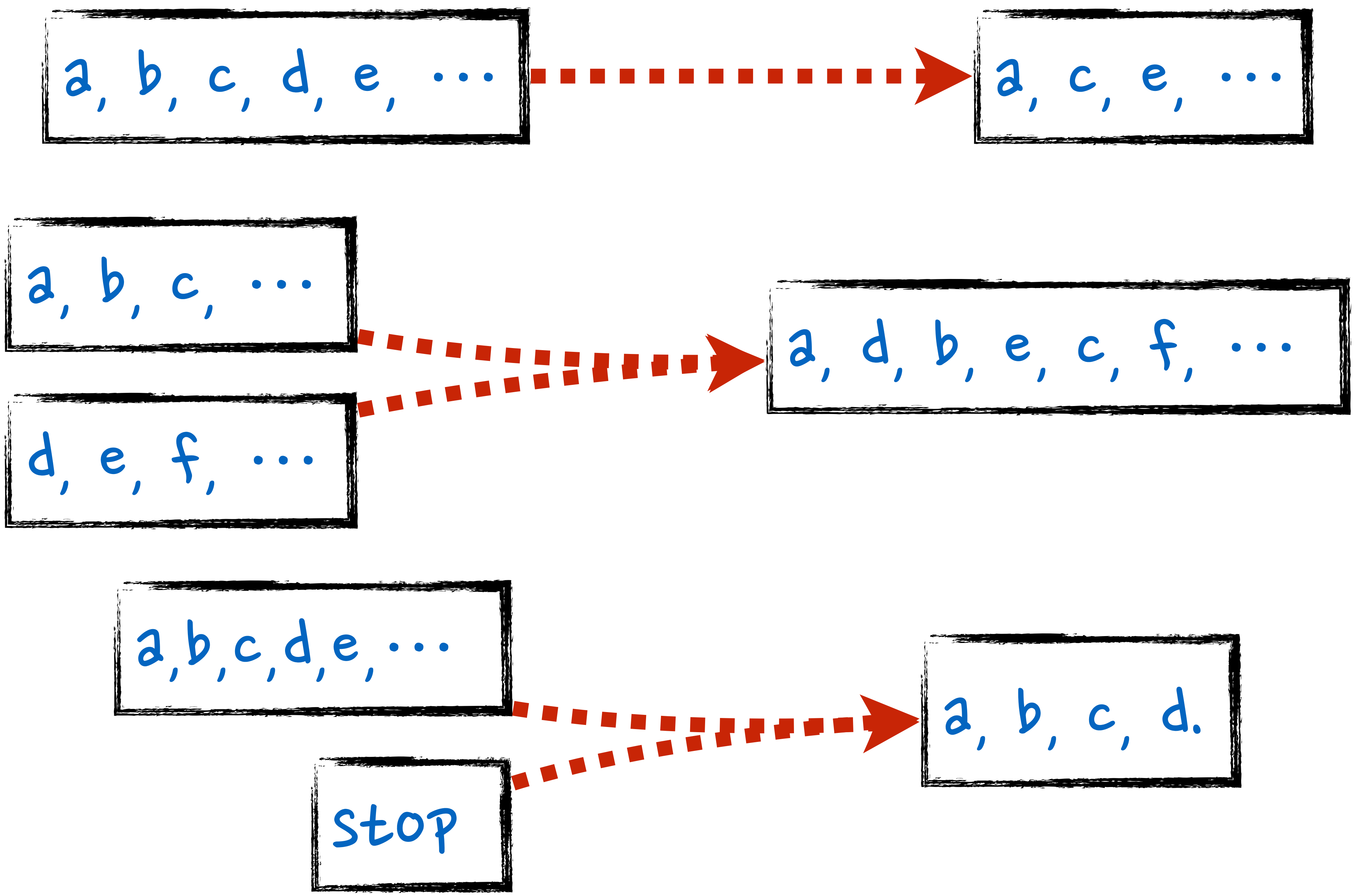


<true*.a|c|d|e>true

[true*.a|c|d|e.e]false

Conn = hide($\{c,d\}$,
block($\{c_1,c_2,d_1,d_2\}$,
comm($\{c_1|c_2 \to c,\ d_1|d_2 \to \mathbf{d}\}$,
Merger || Lossy || FIFO1)))

Build connectors



Can you prove?

colourings and port automata provide equivalent semantics

$$\mathcal{A}(C_1) = (Q_1, \mathcal{N}_1, \rightarrow_1, q_{0,1})$$
$$\mathcal{A}(C_2) = (Q_2, \mathcal{N}_2, \rightarrow_2, q_{0,2})$$

$\mathcal{CT}(C)$ – colouring table of C

$col(q \xrightarrow{P} q')$ – colouring associated to a transition

$$(\langle q_{0,1}, q_{0,2}\rangle \xrightarrow{P} \langle q_1, q_2\rangle) \in \mathcal{A}(C_1) \bowtie \mathcal{A}(C_2)$$
$$\Rightarrow$$
$$col(\langle q_{0,1}, q_{0,2}\rangle \xrightarrow{P} \langle q_1, q_2\rangle) \in \mathcal{CT}(C_1) \bowtie \mathcal{CT}(C_2)$$

Can you prove? (more generically)

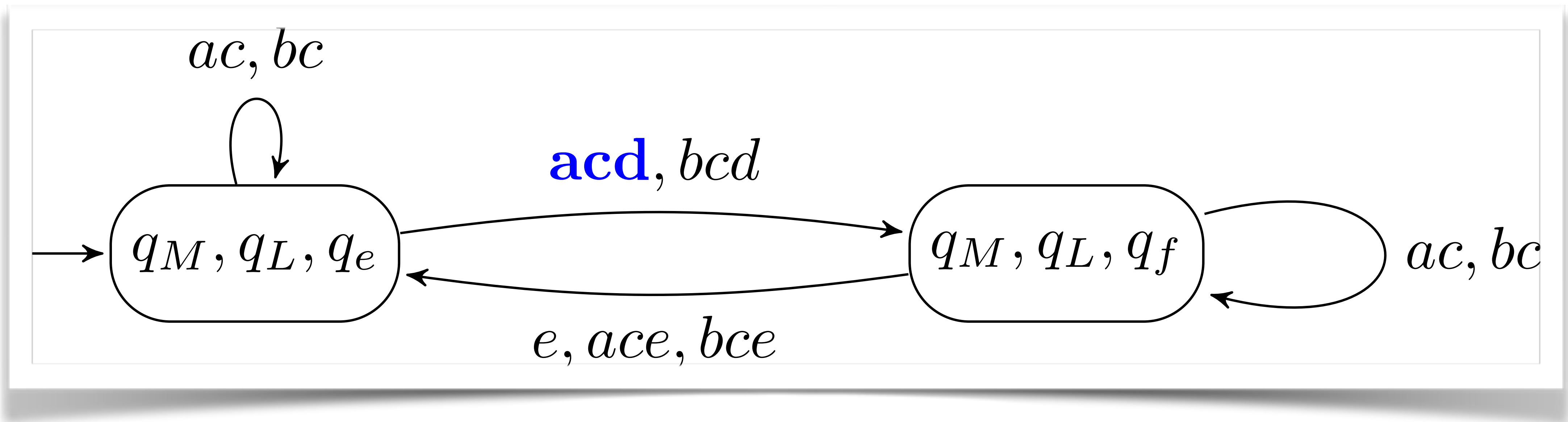
colourings and port automata provide equivalent semantics

$$\mathcal{A} = (\mathcal{Q}, \mathcal{N}, \rightarrow, \{q_0\})$$

$$(q_0 \xrightarrow{P} q) \in \mathcal{A}(C)$$

$$\Rightarrow$$

$$col(P, \mathcal{N}) \;\in\; \mathcal{CT}(C)$$



Constraint Automata

Christel Baier, Marjan Sirjani, Farhad Arbab, Jan Rutten. Modeling Component Connectors in Reo by Constraint Automata. 2004

Constraint Automata

Automata labelled by

- a **data constraint** which represents a set of data assignments to port names

$$g \quad ::= \quad \mathsf{true} \mid d_A = v \mid g_1 \vee g_2 \mid \neg g$$

Over Data and Ports

Note: other constraints, such as

$$d_A = d_B \stackrel{\text{abv}}{=} \vee_{d \in Data}(d_A = d \wedge d_B = d)$$

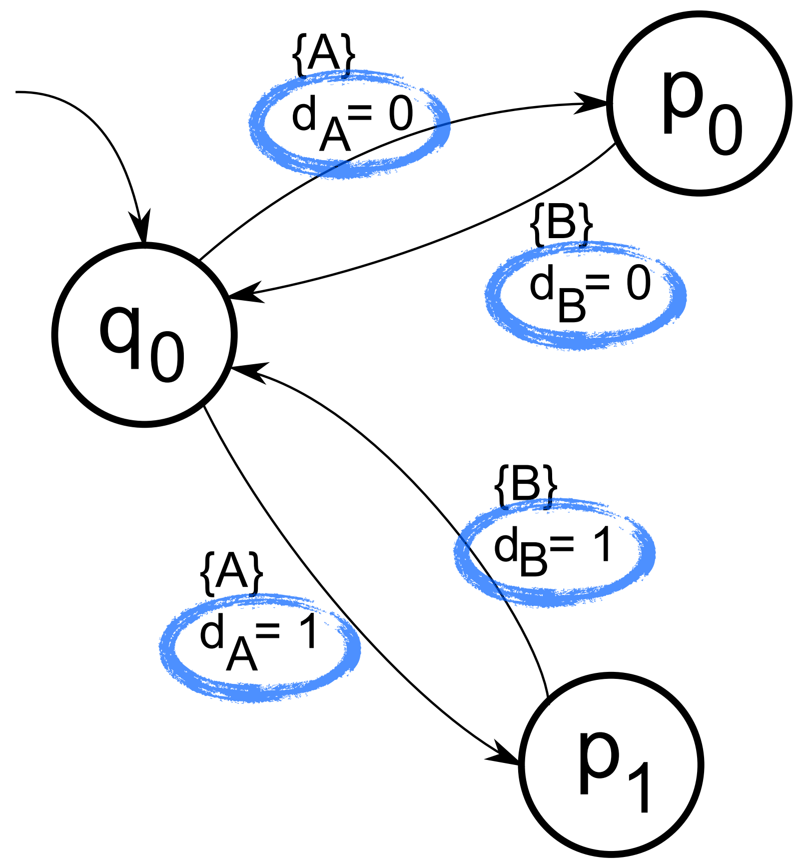
are derived.

- a **name set** which represents the set of port names at which IO can occur

States represent the configurations of the corresponding connector, while transitions encode its maximally-parallel stepwise behaviour.

Constraint Automata

Example: FIFO1



Constraint Automata - Definition

$\mathcal{A} = (\mathcal{Q}, \mathcal{N}, \rightarrow, \mathcal{Q}_0)$

$\mathcal{Q}$	set of states
$\mathcal{N}$	a set of ports $\mathcal{N}$
$\mathcal{Q}_0 \subseteq \mathcal{Q}$	a set of initial states
$\rightarrow \ \subseteq \mathcal{Q} \times 2^{\mathcal{N}} \times DC \times \mathcal{Q}$	a transition relation such that $\xrightarrow{P,g}$ iff

1. $P \neq \emptyset$
2. $g \in DC(P, Data)$

($DC(P, Data)$ is the set of data constraints over Data and P)

Constraint Automata - Definition

$s \xrightarrow{P,g} s'$ iff

1. $P \neq \emptyset$
2. $g \in DC(P, Data)$

in configuration s, ports in P can perform IO operations which meet guard g and lead to s'

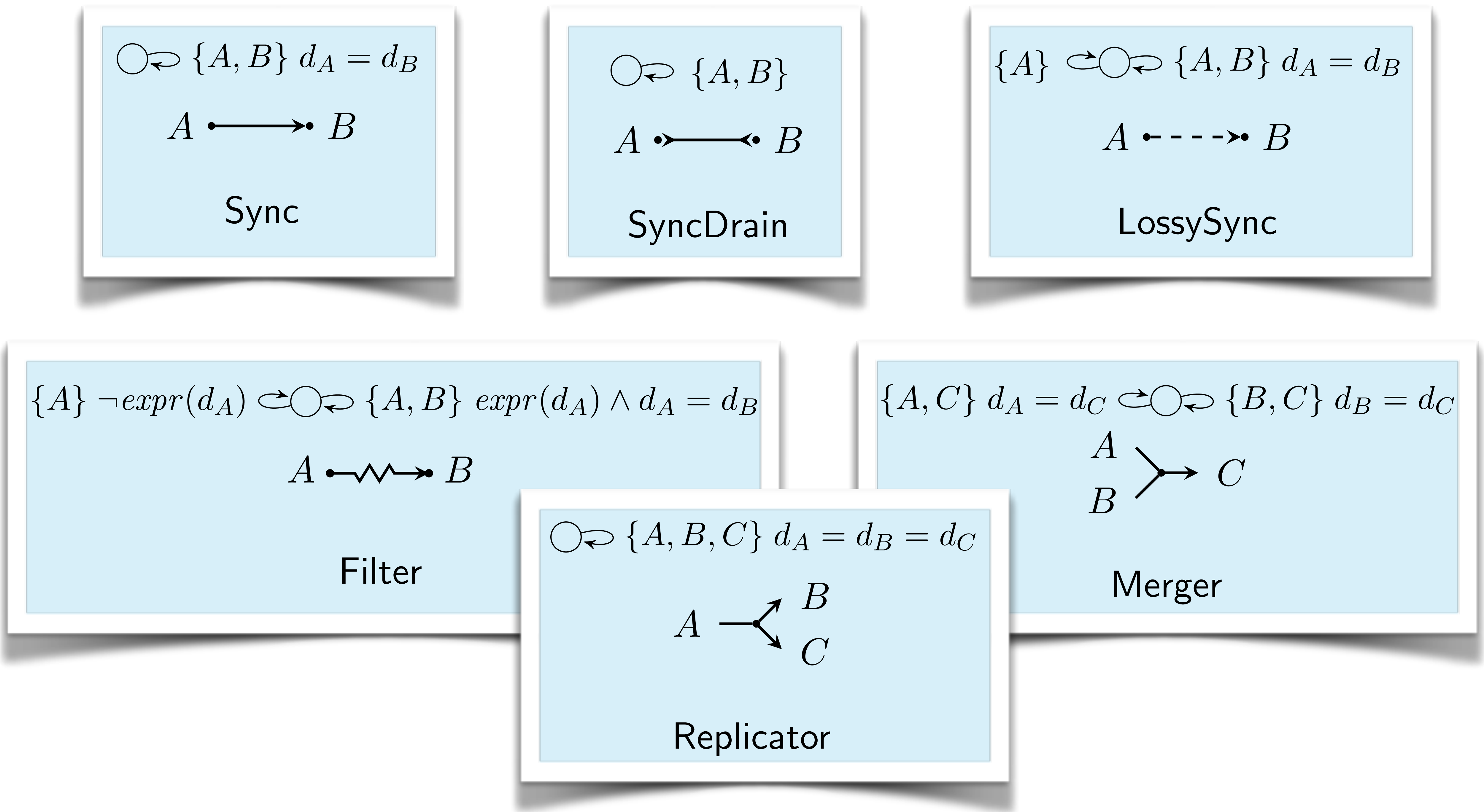
transitions fire only if data occurs at a (set of) ports P

behaviour depends only on observed data (not on future evolution)

Constraint Automata as a semantics for Reo

- cannot capture context-awareness [Baier, Sirjani, Arbab, Rutten 2006], but forms the basis for more elaborated models (eg, Reo automata)
- captures all behaviour alternatives of a connector; useful to generate a state-machine implementing the connector's behaviour
- basis for several tools, including the model checker Vereofy [Kluppelholz, Baier 2007]

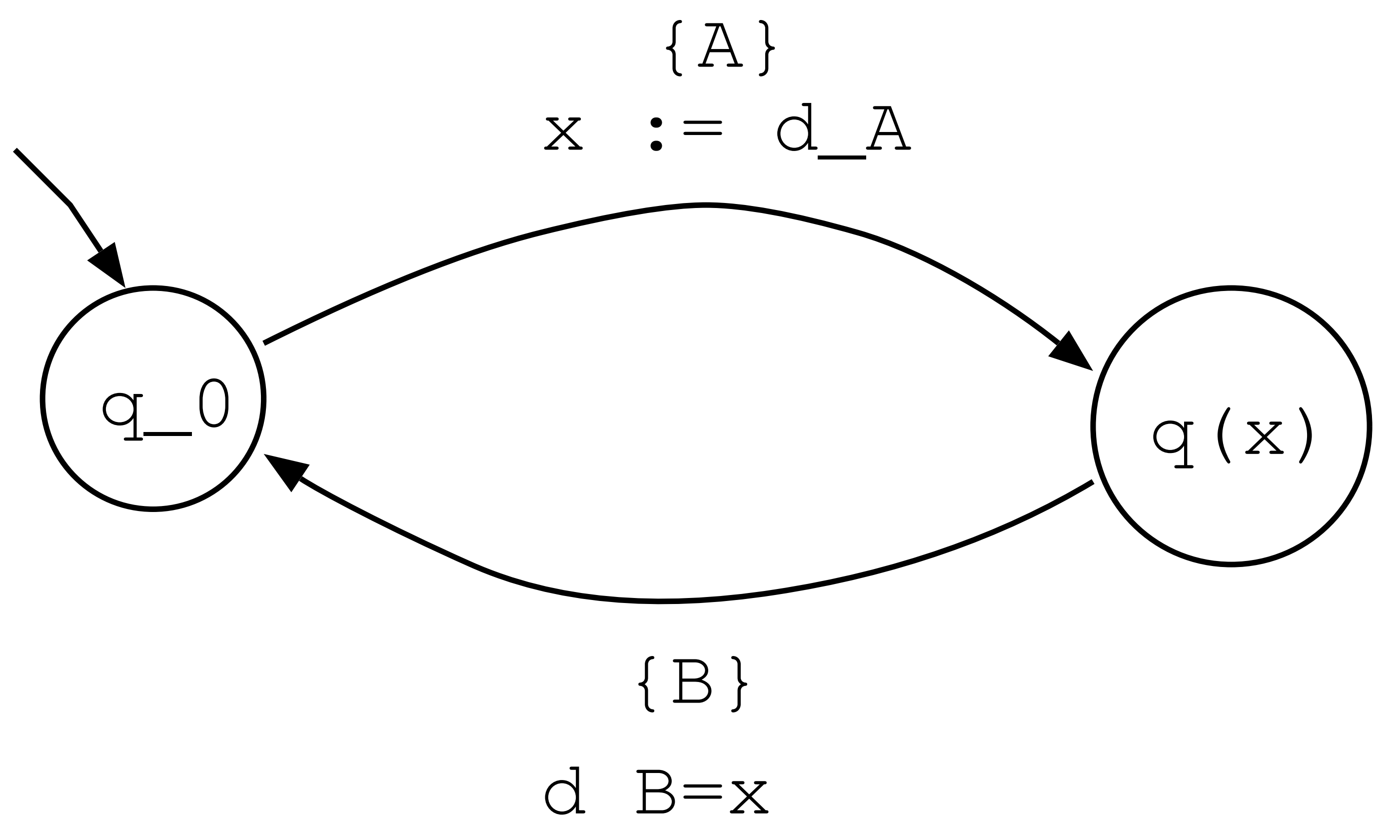
Constraint Automata - Reo connectors



[Parameterised constraint automata]

States are parametric on data values ... therefore capturing complex constraint automata emerging form data-dependencies

Example: 1 bounded FIFO



[Parameterised constraint automata]

$\mathcal{L} = (\mathcal{L}, \mathcal{N}, \mathcal{V}, \rightarrow, \mathcal{L}_0, init)$

$\mathcal{L}$	set of locations
$\mathcal{N}$	a set of ports
$\mathcal{V}$	a set of variables
$\mathcal{L}_0 \subseteq \mathcal{L}$	a set of initial states
$init$	an initialisation of variables
$\rightarrow \ \subseteq \mathcal{L} \times 2^{\mathcal{N}} \times DC \times Assgn \times \mathcal{L}$	a transition relation such that $\xrightarrow{P,g,h}$ iff

1. $P \neq \emptyset$
2. $g \in DC(P, Data, \mathcal{V})$
3. $h \in \mathcal{V} \rightarrow Expr(P, Data, \mathcal{V})$

($DC(P, Data, \mathcal{V})$ is the set of data constraints ove Data, P, and $\mathcal{V}$)
($Expr(P, Data, \mathcal{V})$ is an expression over Data, P, and $\mathcal{V}$)

[Parameterised constraint automata]

$\mathcal{L} = (\mathcal{L}, \mathcal{N}, \mathcal{V}, \to, \mathcal{L}_0, init)$

Like clocks

Like clock constraints

$\mathcal{L}$	set of locations
$\mathcal{N}$	a set of ports
$\mathcal{V}$	a set of variables
$\mathcal{L}_0 \subseteq \mathcal{L}$	a set of initial states
$init$	an initialisation of variables
$\to \ \subseteq \mathcal{L} \times 2^{\mathcal{N}} \times DC \times Assgn \times \mathcal{L}$	a transition relation such that $\xrightarrow{P,g,h}$ iff

1. $P \neq \emptyset$
2. $g \in DC(P, Data, \mathcal{V})$
3. $h \in \mathcal{V} \to Expr(P, Data, \mathcal{V})$

Like clock updates

($DC(P, Data, \mathcal{V})$ is the set of data constraints ove Data, P, and $\mathcal{V}$)
($Expr(P, Data, \mathcal{V})$ is an expression over Data, P, and $\mathcal{V}$)

Composing constraint automata

Definition 4.1 [*Product-automaton*] The product-automaton of the two constraint automata $\mathcal{A}_1 = (Q_1, \mathcal{N}ames_1, \longrightarrow_1, Q_{0,1})$ and $\mathcal{A}_2 = (Q_2, \mathcal{N}ames_2, \longrightarrow_2, Q_{0,2})$, is:

$$\mathcal{A}_1 \bowtie \mathcal{A}_2 \;=\; (Q_1 \times Q_2, \mathcal{N}ames_1 \cup \mathcal{N}ames_2, \longrightarrow, Q_{0,1} \times Q_{0,2})$$

where $\longrightarrow$ is defined by the following rules:

$$\frac{q_1 \xrightarrow{N_1, g_1}_1 p_1, \quad q_2 \xrightarrow{N_2, g_2}_2 p_2, \quad N_1 \cap \mathcal{N}ames_2 = N_2 \cap \mathcal{N}ames_1}{\langle q_1, q_2 \rangle \xrightarrow{N_1 \cup N_2, g_1 \wedge g_2} \langle p_1, p_2 \rangle}$$

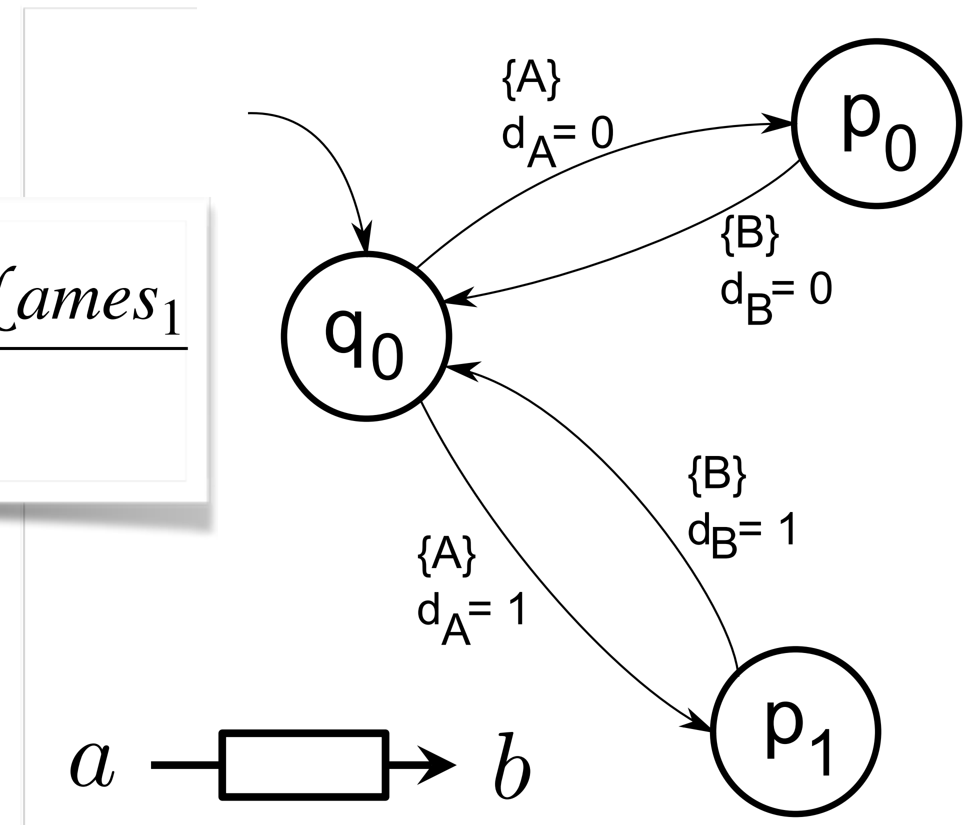
and

$$\frac{q_1 \xrightarrow{N, g}_1 p_1, \quad N \cap \mathcal{N}ames_2 = \emptyset}{\langle q_1, q_2 \rangle \xrightarrow{N, g} \langle p_1, q_2 \rangle}$$

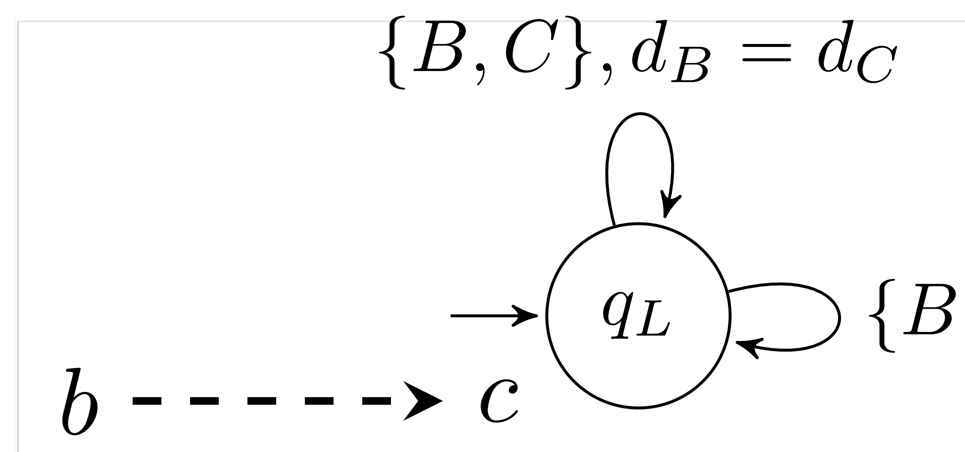
Formalize and compose

$$\frac{q_1 \xrightarrow{N_1, g_1}_1 p_1, \quad q_2 \xrightarrow{N_2, g_2}_2 p_2, \quad N_1 \cap \mathcal{N}ames_2 = N_2 \cap \mathcal{N}ames_1}{\langle q_1, q_2 \rangle \xrightarrow{N_1 \cup N_2, g_1 \wedge g_2} \langle p_1, p_2 \rangle}$$

$$\frac{q_1 \xrightarrow{N, g}_1 p_1, \quad N \cap \mathcal{N}ames_2 = \emptyset}{\langle q_1, q_2 \rangle \xrightarrow{N, g} \langle p_1, q_2 \rangle}$$

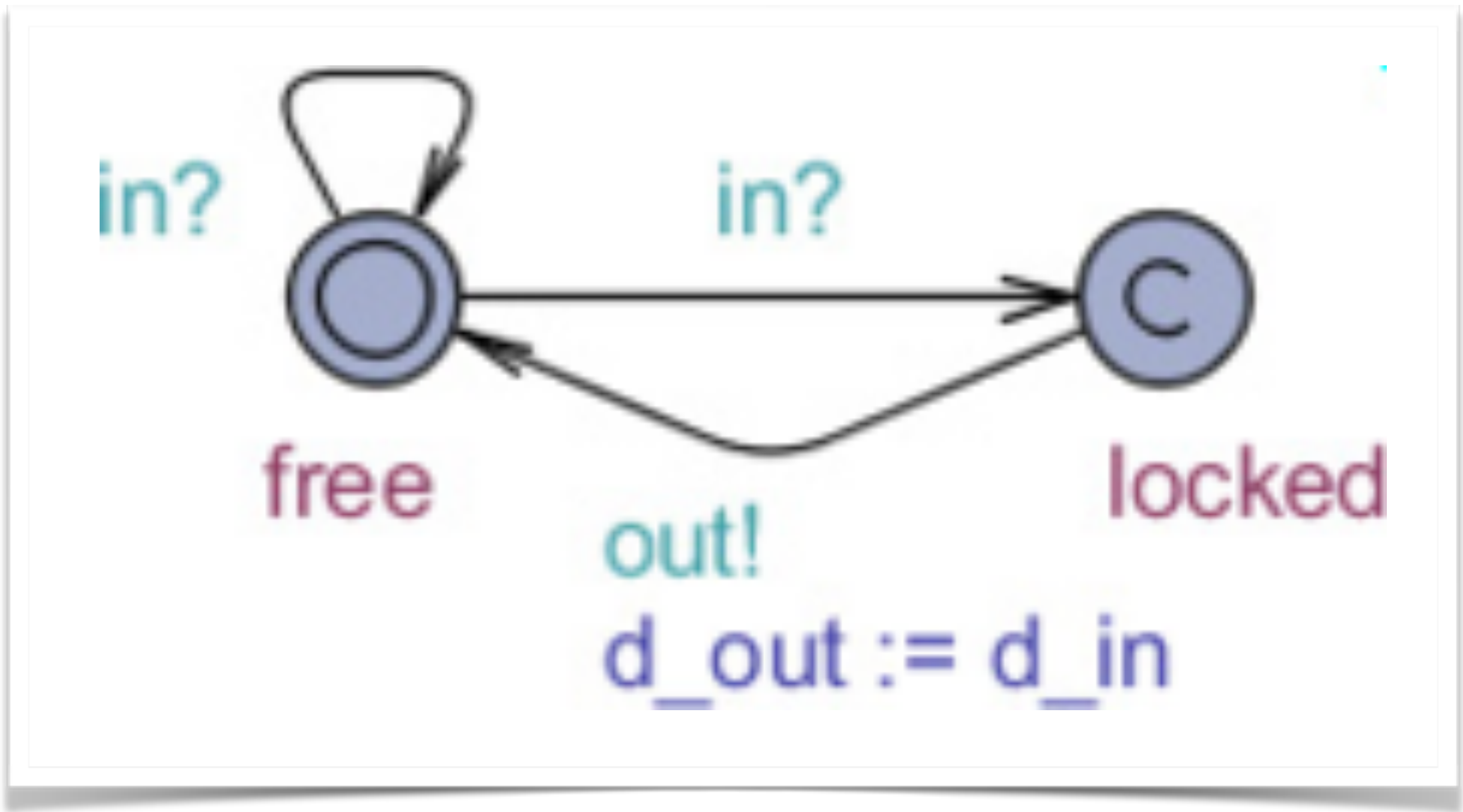


$\bowtie$



You are here

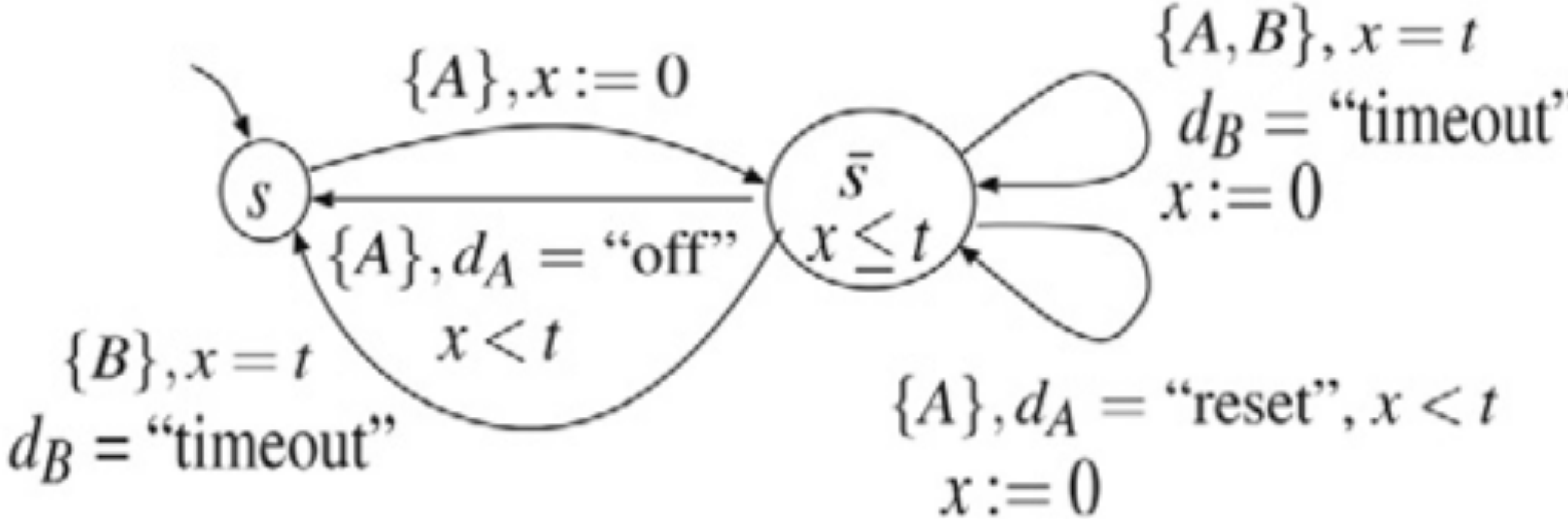
Formalism	Synchr.	Data	Time	Context	Partial
Connector Colouring	CC2	-		**CC3**	-
Automata	Port Automata	**Constraint Automata**	**Time CA**	-	-
Constraints	✓	✓	✗	✓	✓



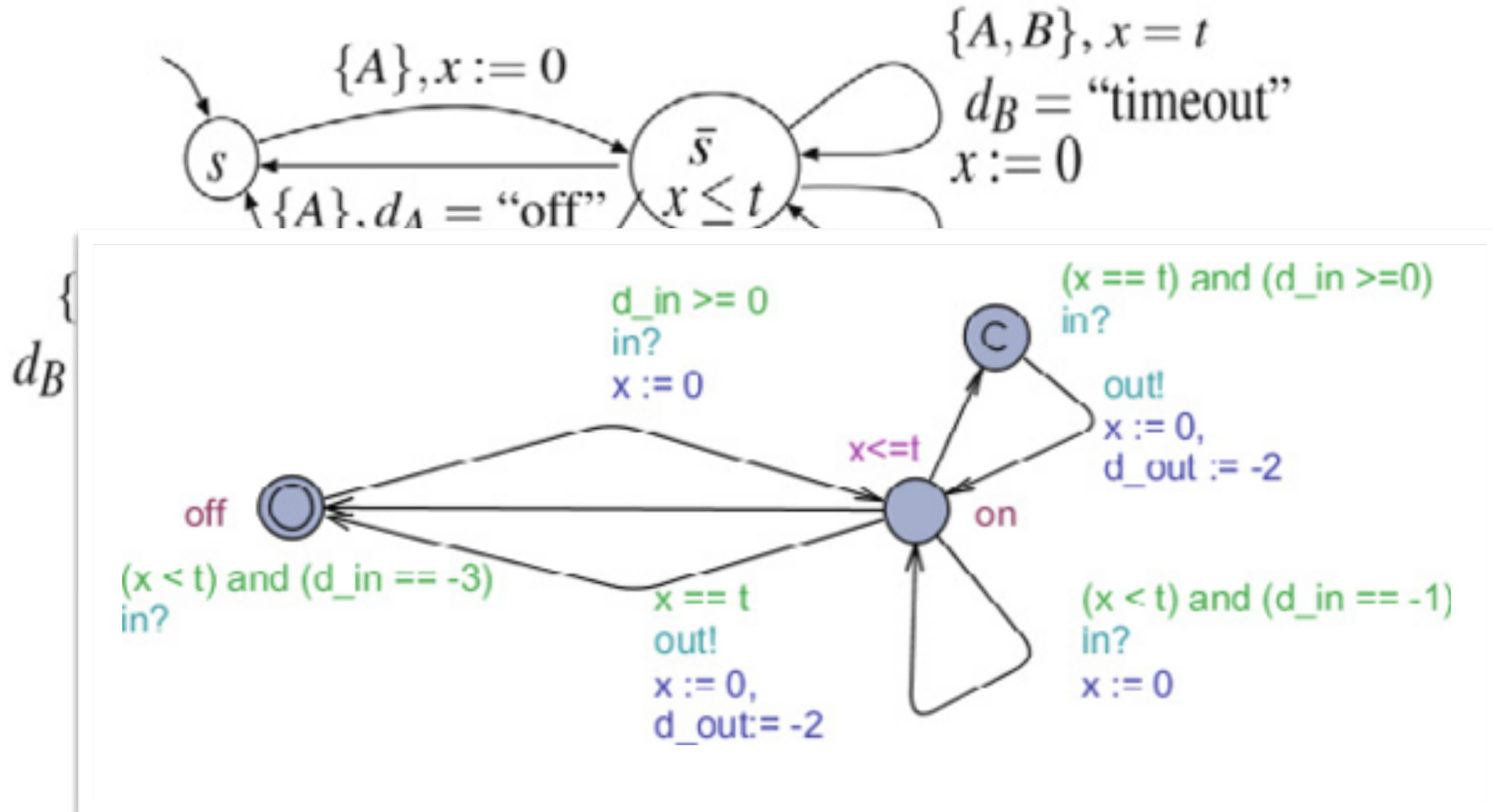
Timed Port/Const. Automata

Natallia Kokash, Mohammad Mahdi Jaghoori, Farhad Arbab.
From Timed Reo Networks to Networks of Timed Automata. 2013

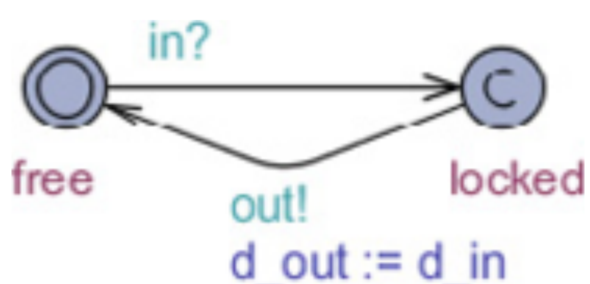
Timer (Data) Channel



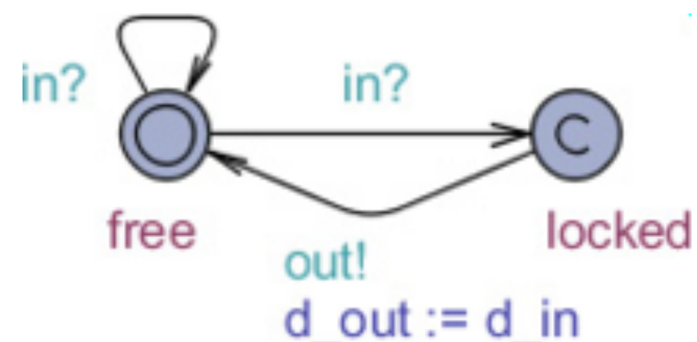
Timer (Data) Channel



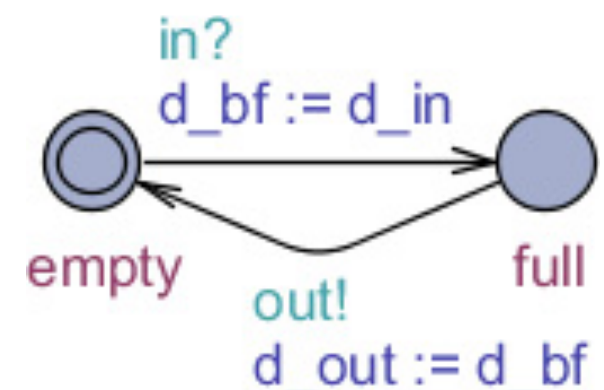
Time extension



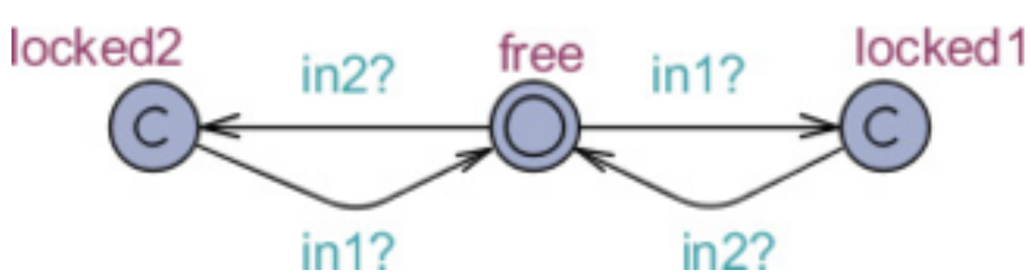
Sync



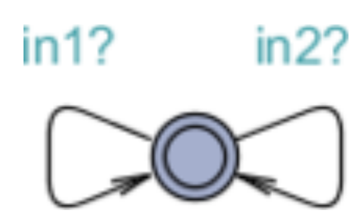
LossySync



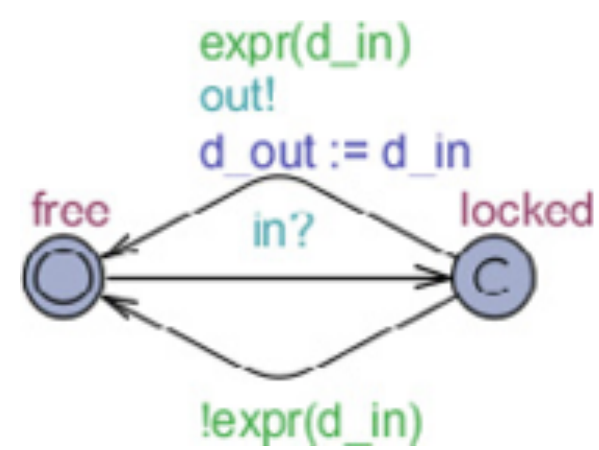
FIFO



SyncDrain



AsyncDrain



Filter

Formally

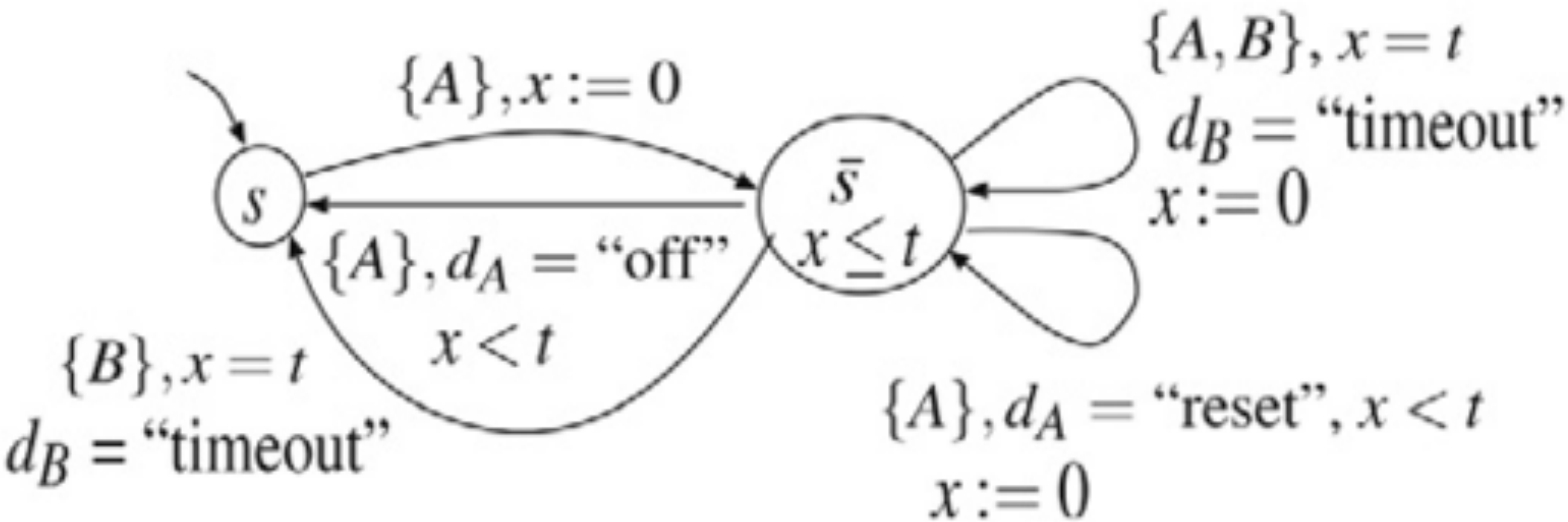
Definition 2.1 [Constraint automaton (CA)] A constraint automaton $\mathcal{A} = (S, \mathcal{N}, \rightarrow, s_0)$ consists of a set of states (also called locations) S, a set of port names $\mathcal{N}$, a transition relation $\rightarrow \subseteq S \times 2^{\mathcal{N}} \times DC \times S$, where DC is the set of data constraints over a finite data domain $Data$, and an initial state $s_0 \in S$.

Definition 2.2 [Timed constraint automaton (TCA) [3]] A TCA is an extended constraint automaton $\mathcal{A} = (S, \mathcal{N}, \rightarrow, s_0, \mathcal{C}, ic)$ with transition relation $\rightarrow \subseteq S \times 2^{\mathcal{N}} \times DC \times CC \times 2^{\mathcal{C}} \times S$ such that $\mathcal{C}$ is a finite set of clocks and $ic : S \rightarrow CC$ is a function that assigns a clock constraint, called an invariance condition $ic(s)$ to each location s of $\mathcal{A}$.

Natallia Kokash, Mohammad Mahdi Jaghoori, Farhad Arbab.
From Timed Reo Networks to Networks of Timed Automata. 2013

without data!

Build channels (TCA)



delay(t) - wait more than "t" time (exclusive).

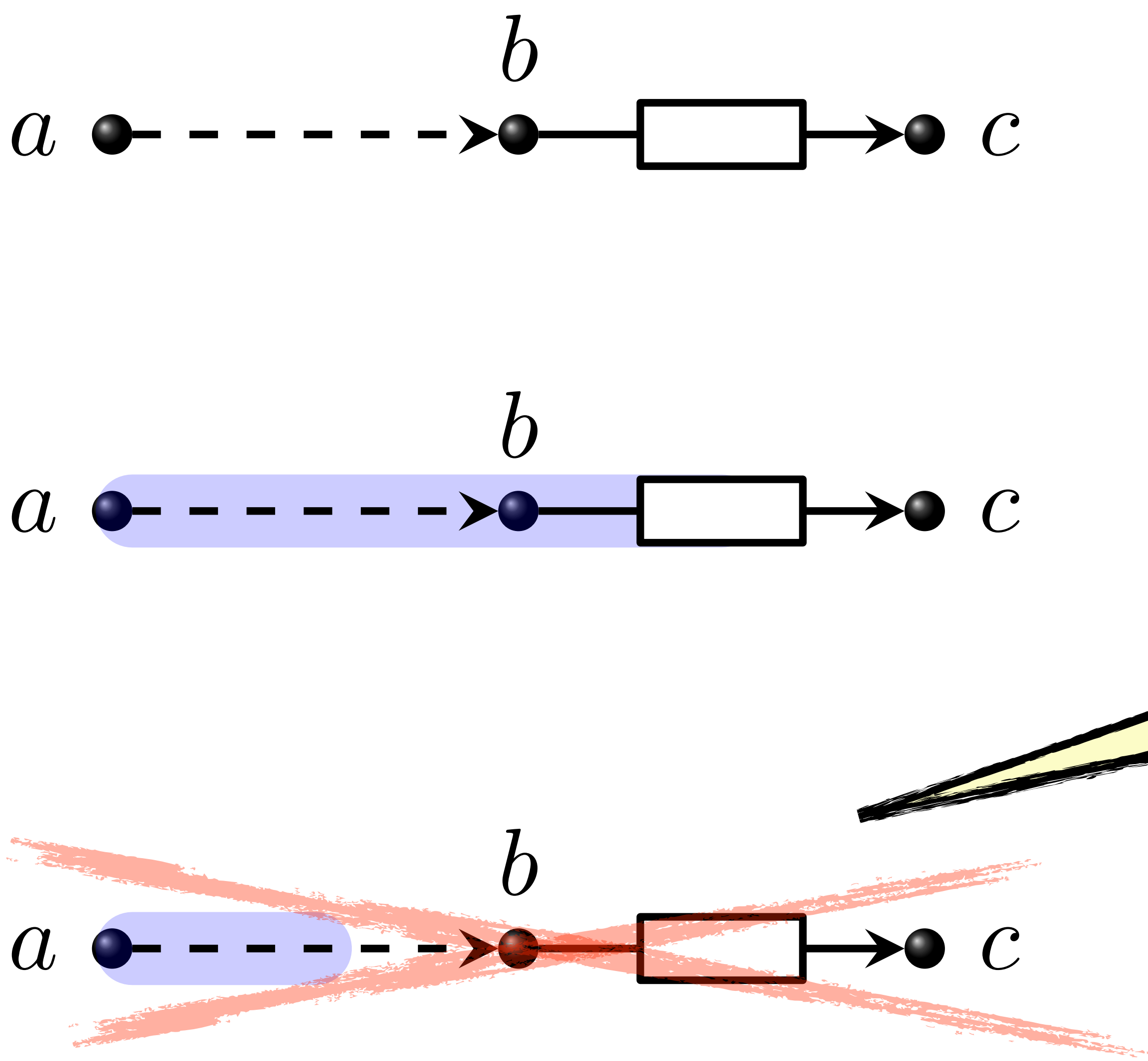
delayedTimer(t) - wait between "tmin" and "tmax", inclusive.

timer(t) - with "off" and "reset" ports.

You are here

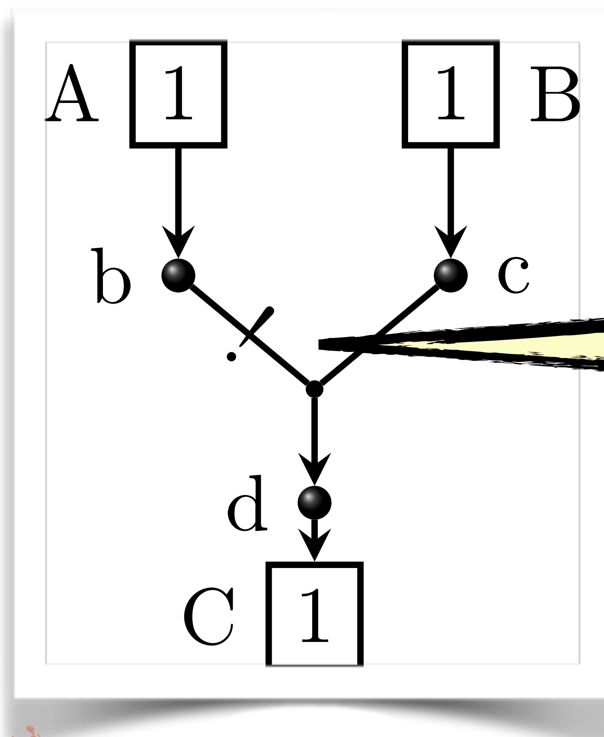
Formalism	Synchr.	Data	Time	Context	Partial
Connector Colouring	CC2	-		**CC3**	-
Automata	Port Automata	**Constraint Automata**	**Time CA**	-	-
Constraints	✓	✓	✗	✓	✓

2 reasons for context

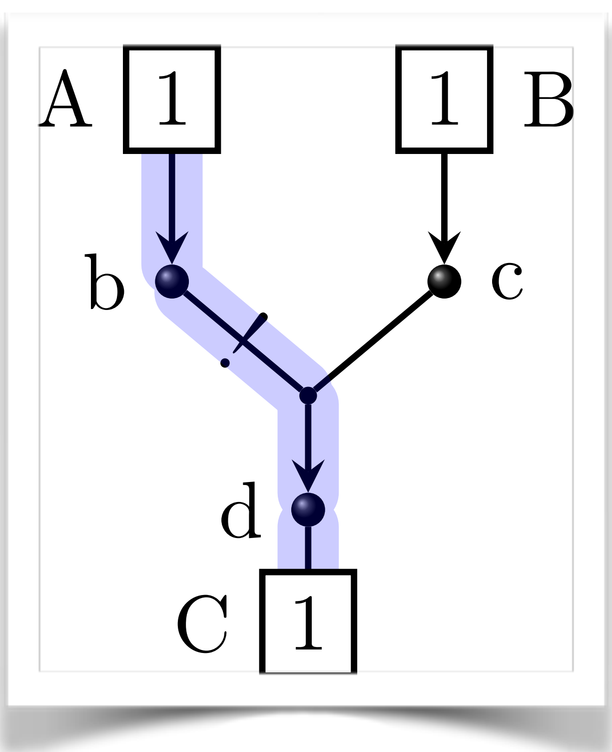


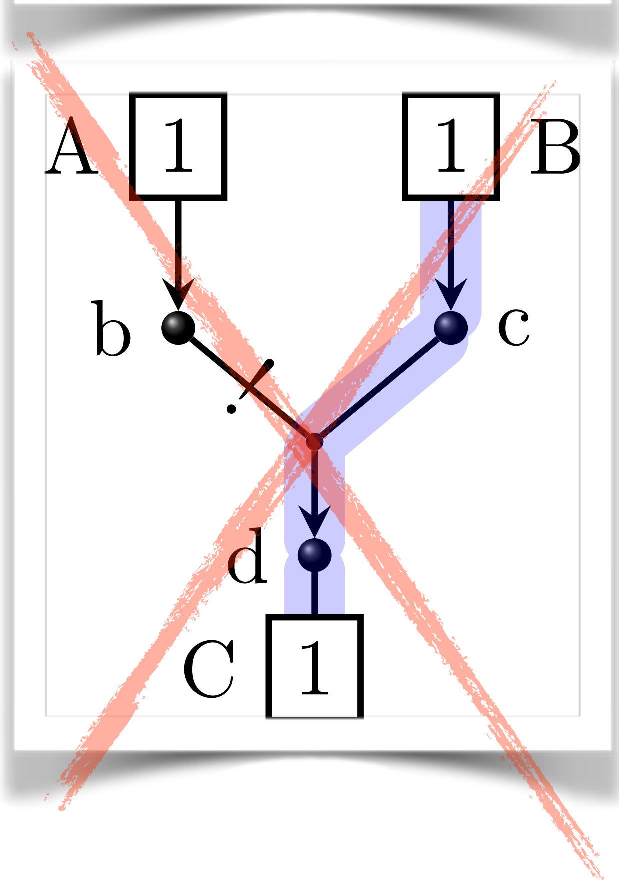
1 - avoid data loss when the context (FIFO) can receive the data.

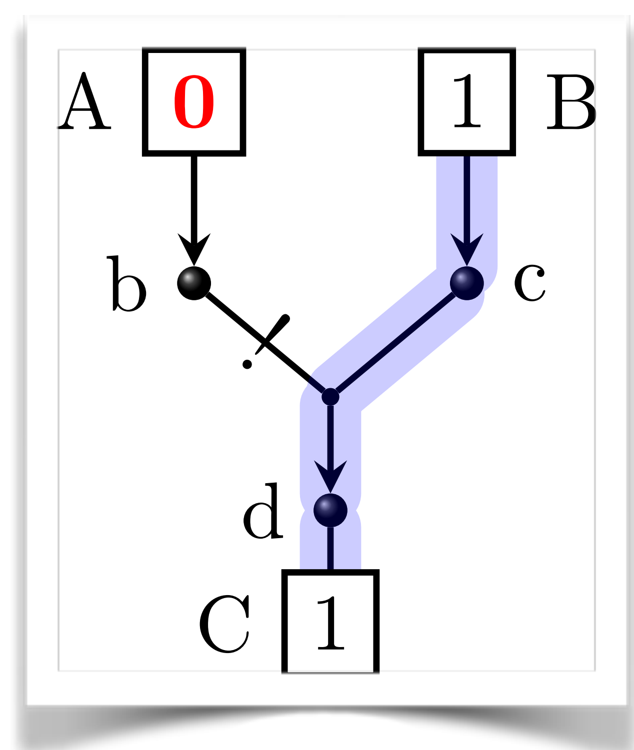
2 reasons for context



2 - give priority based on the context (writer)





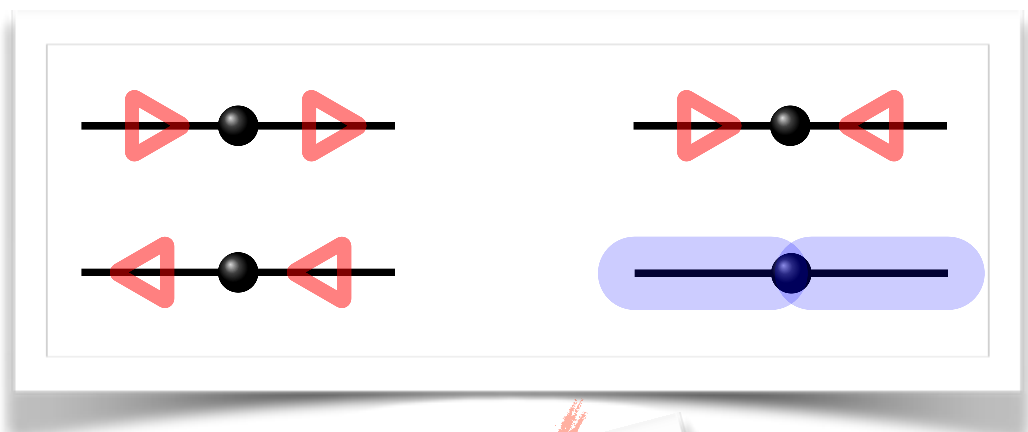


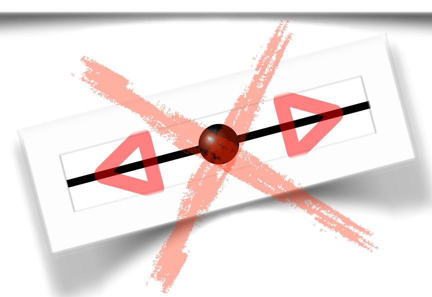
Context = 3 colours

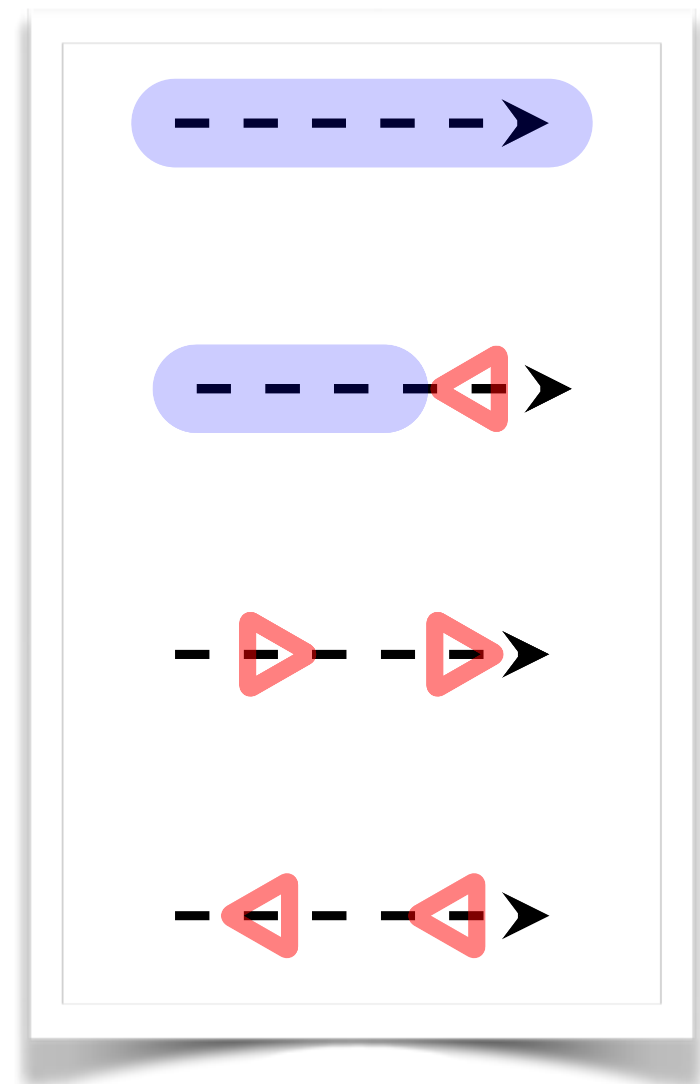
- *Colouring*:

  End → {Flow, GiveReason, GetReason}

- *Composition* = matching colours:





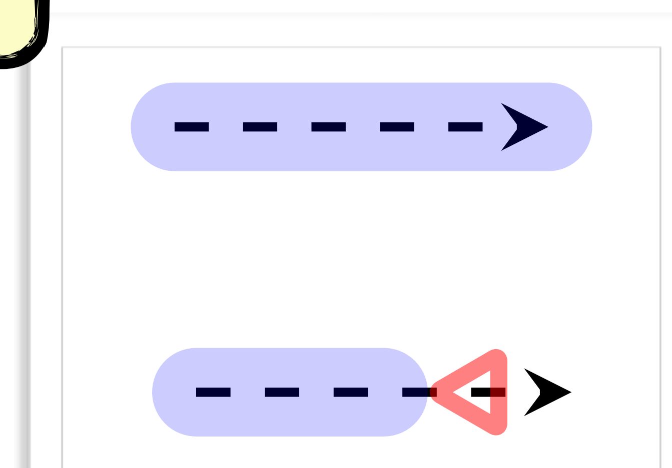


Context = 3 colours

- *Colo*

$$\text{End} = \{e_1, \ldots, e_n\} \cup \{\overline{e_1}, \ldots, \overline{e_n}\}$$

End → {Flow, GiveReason,GetReason}



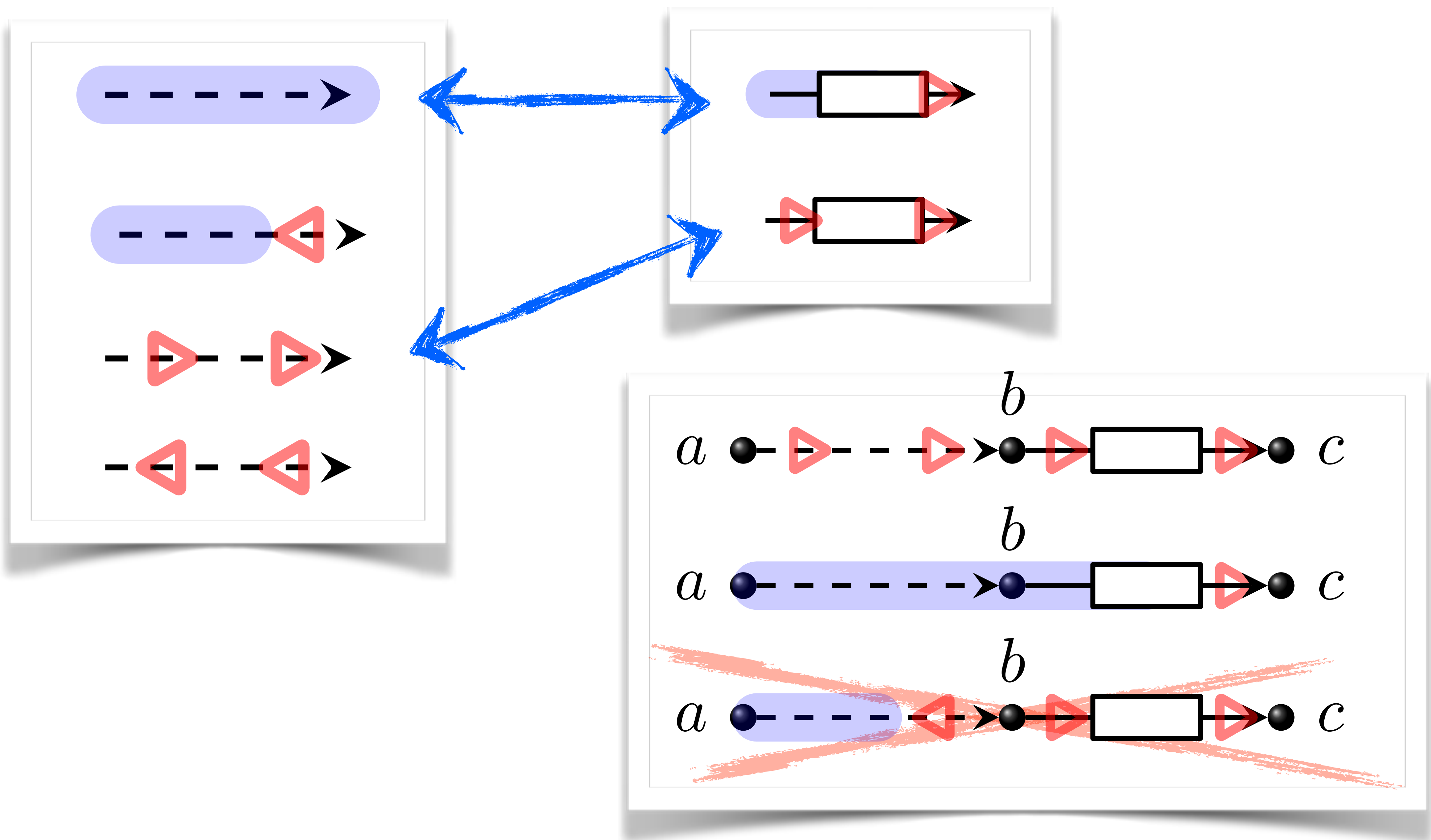
- *Composition* = matching colours:

$$
\begin{aligned}
CT_1 \bowtie CT_2 \quad &= \\
&\{cl_1 \bowtie cl_2 \mid cl_1 \in CT_1, cl_2 \in CT_2, cl_1 \frown cl_2\}
\end{aligned}
$$

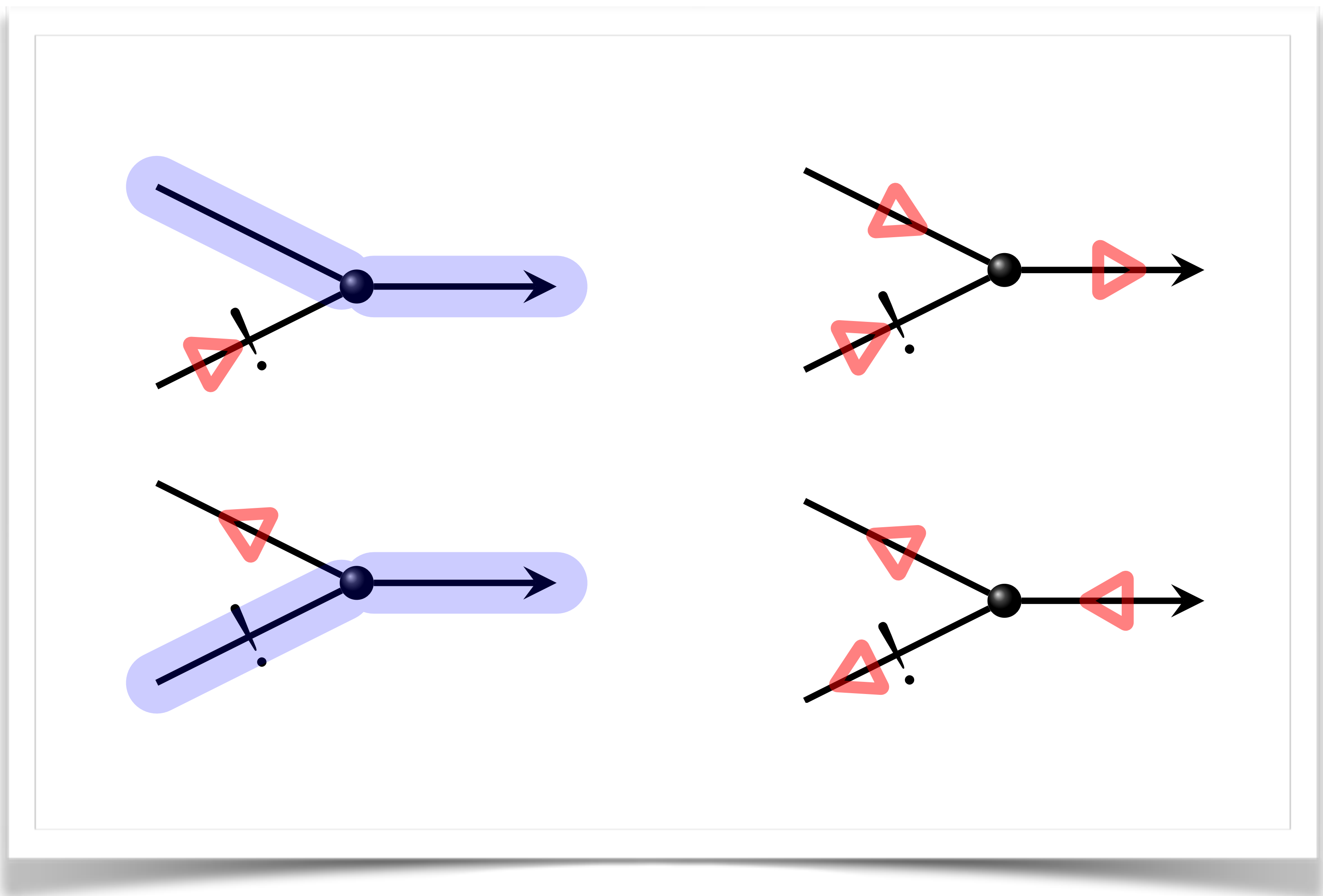
$$
\begin{aligned}
cl_1 \frown cl_2 = \forall e_1 \in \mathsf{dom}(cl_1) \cdot \forall e_2 \mathsf{dom}(cl_2) \cdot \\
e_1 = \overline{e}_2 \Rightarrow \\
(cl_1(e), cl_2(e)) \in \{(\triangleright, \triangleright), (\triangleleft, \triangleleft), (\triangleright, \triangleleft), \}
\end{aligned}
$$

$$cl_1 \bowtie cl_2 = cl_1 \cup cl_2$$

Composition



Priority with 3 colours



Connector colouring 3

- Compositional – composition operation is associative, commutative, and does not require post-processing.
- *Reasons* for the absence of flow are propagated.
- Expresses priority.
- 2 colours $\Leftrightarrow$ constraint automata (without data)
- 3 colours: + expressive ($\Leftrightarrow$ intentional automata)

Build a connector

prefer fast FIFO