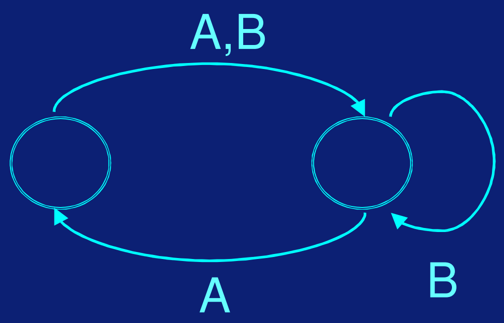
CIC 2009, Braga



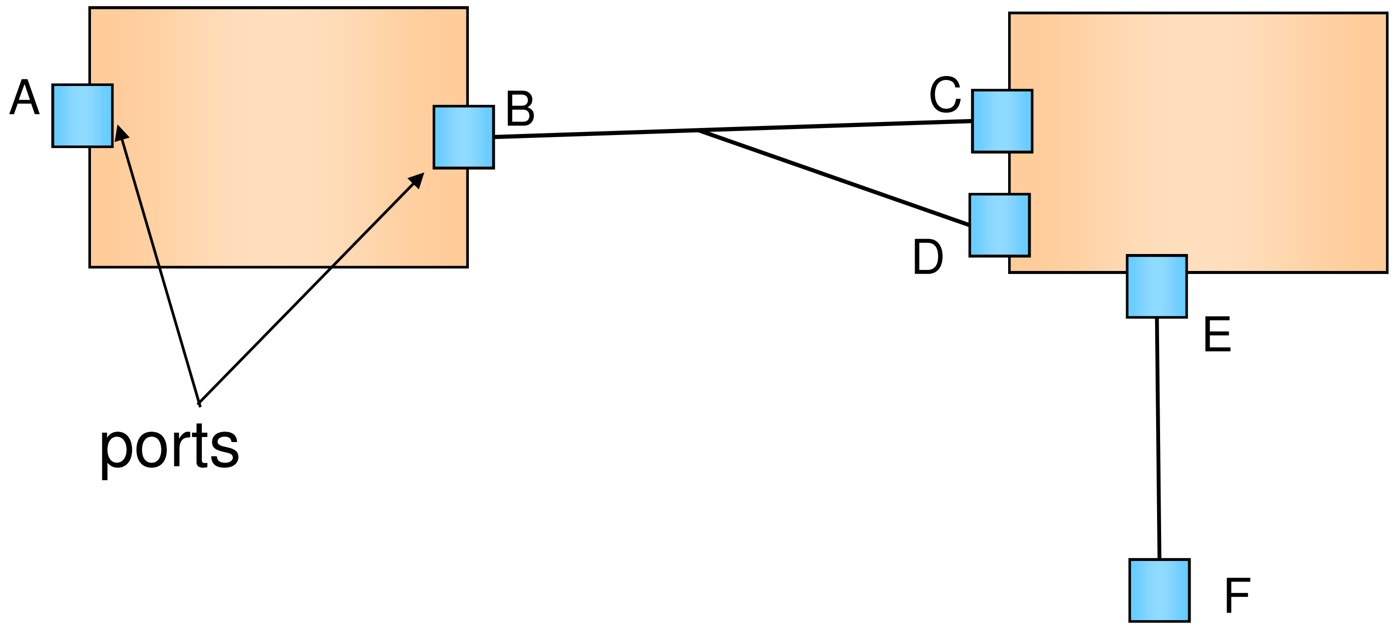
Automata models of component connectors

Marcello Bonsangue
Dave Clarke
Mohammad Izadi
Alexandra Silva

Leiden Institute of Advanced Computer Science
Research & Education

Component Connectors

- Component = Unit of computation
- Connector = Unit of interaction



Reo, some connectors

Sync

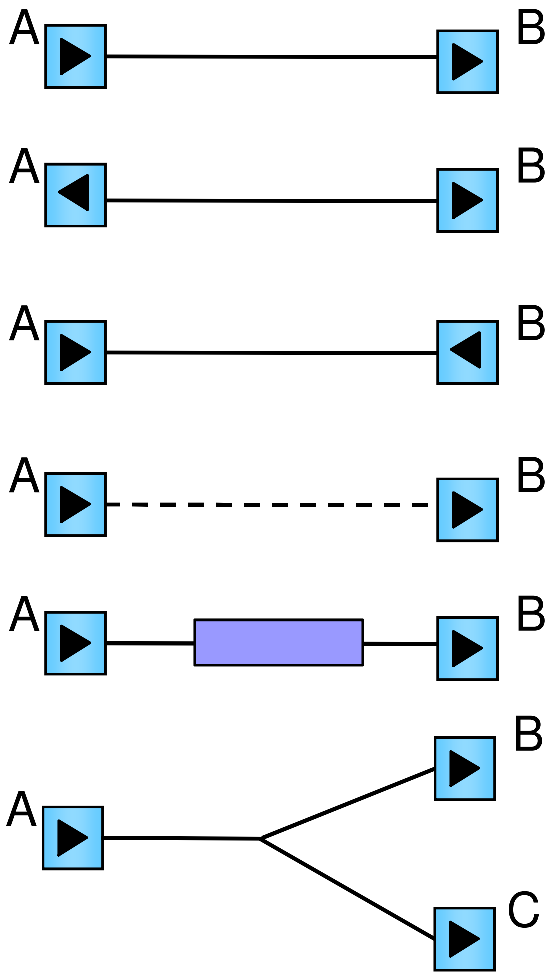
Sync Spout

Sync Drain

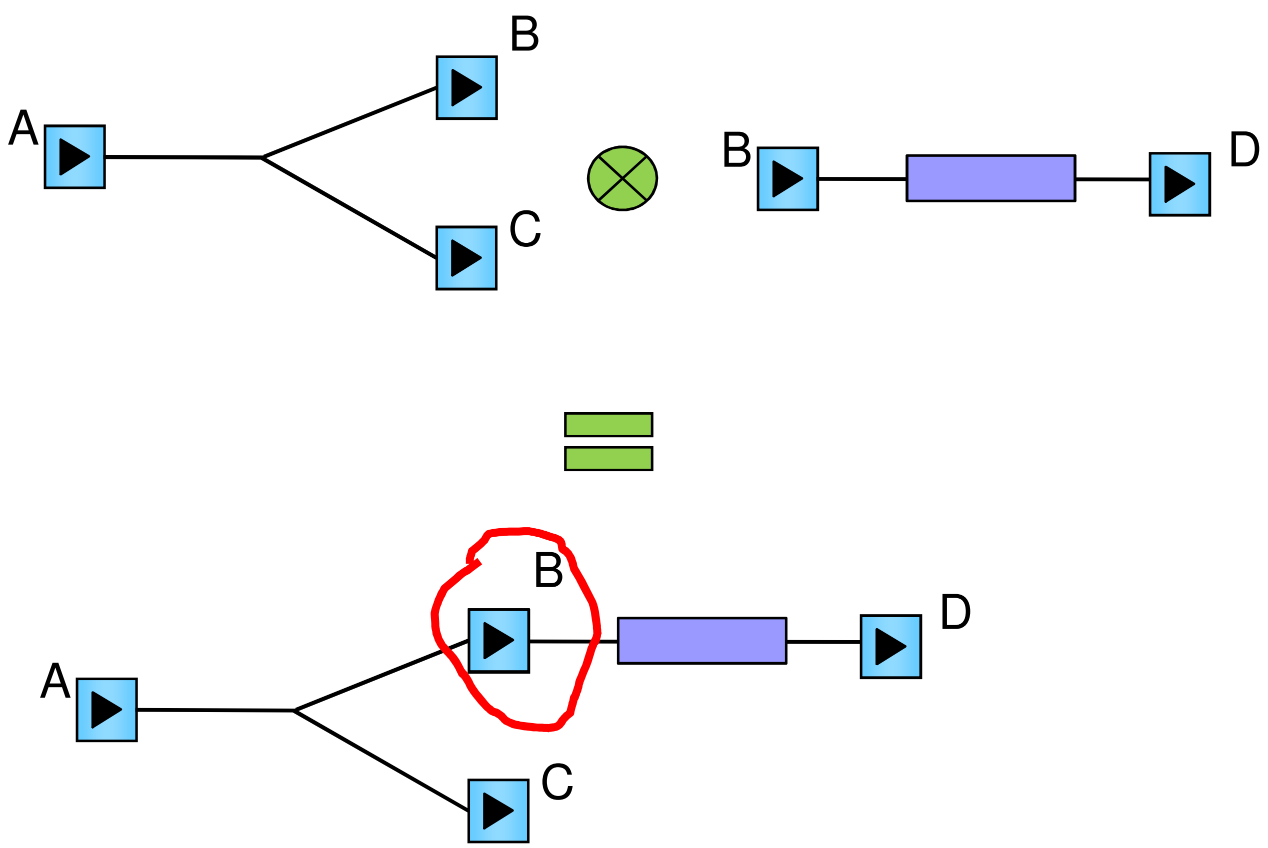
Lossy Sync

FIFO1

Merger



Reo, connector composition



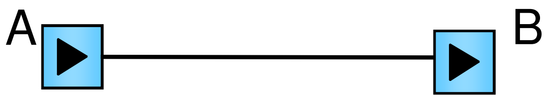
Overview

- Past [2002-2007]
- Present [end 2008 - mid 2009]
- Future [june 2009 -]

Timed Data Strings

- The mother of all Reo semantics

- Connectors are relations of streams of data flow and observation time at each port

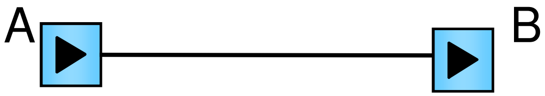


A	h	a	l	l	o		w	o	r	l	
	3	3.5	4.2	4.3	5	5.1	5.2	6	6.8	8	…

B	h	a	l	l	o		w	o	r	l	
	3	3.5	4.2	4.3	5	5.1	5.2	6	6.8	8	…

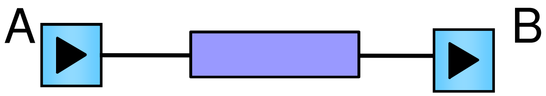
TDS, some connectors

- Sync



  - $A.\delta(0) = B.\delta(0)$ and $A.\tau(0) = B.\tau(0)$
  - A' Sync B'

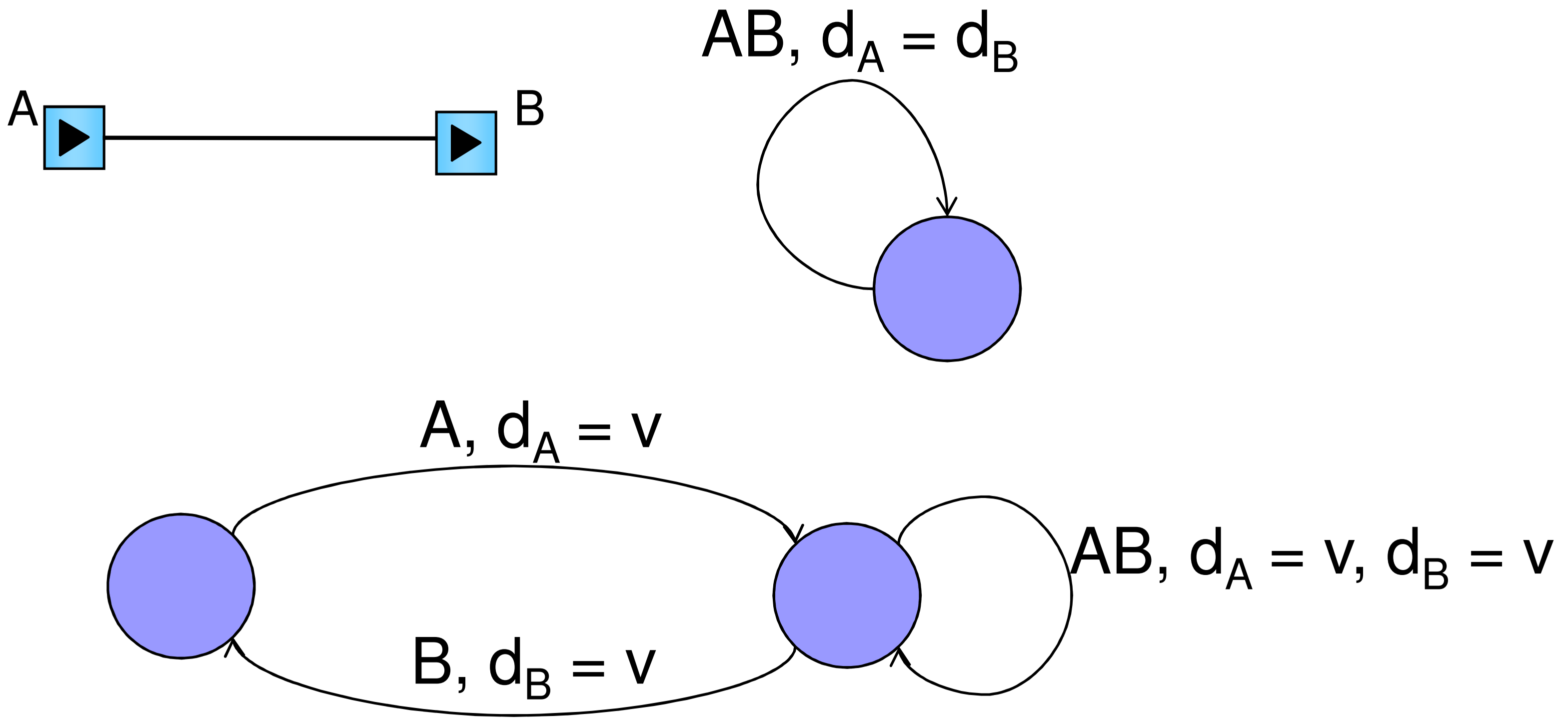
- FIFO1



  - $A.\delta(0) = B.\delta(0)$ and $A.\tau(0) < B.\tau(0) \leq A'.\tau(0)$
  - A' FIFO1 B'

Constraint Automata

- Operational model to describe the behavior of Reo circuits



CA and TDS: where is the time?

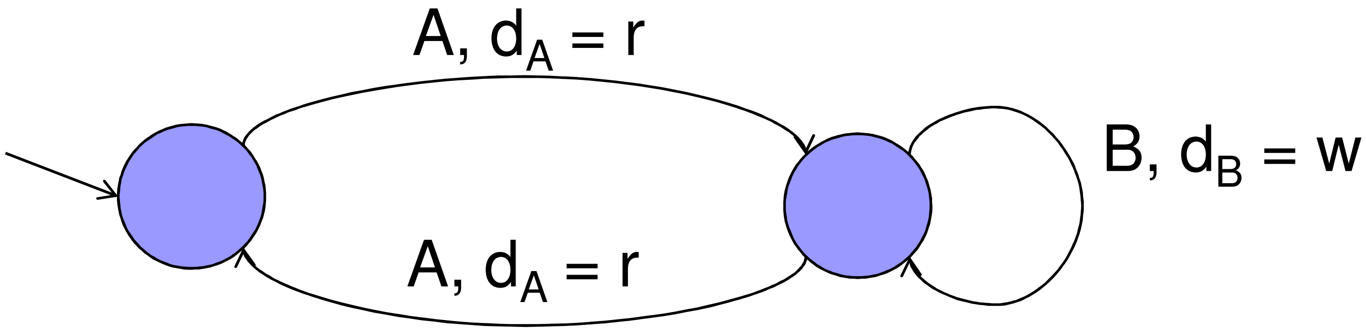
- CAs are acceptors of TDSs

$\theta \in L(\mathcal{A},q)$ iff there exists $q \xrightarrow{N,g} q'$ such that

- $\theta.ports(0) = N$
- $\theta.data(0)$ satisfies the data constraint g
- $\theta' \in L(q')$

where θ.ports is the stream of sets of ports for which a data item is observed at same time.

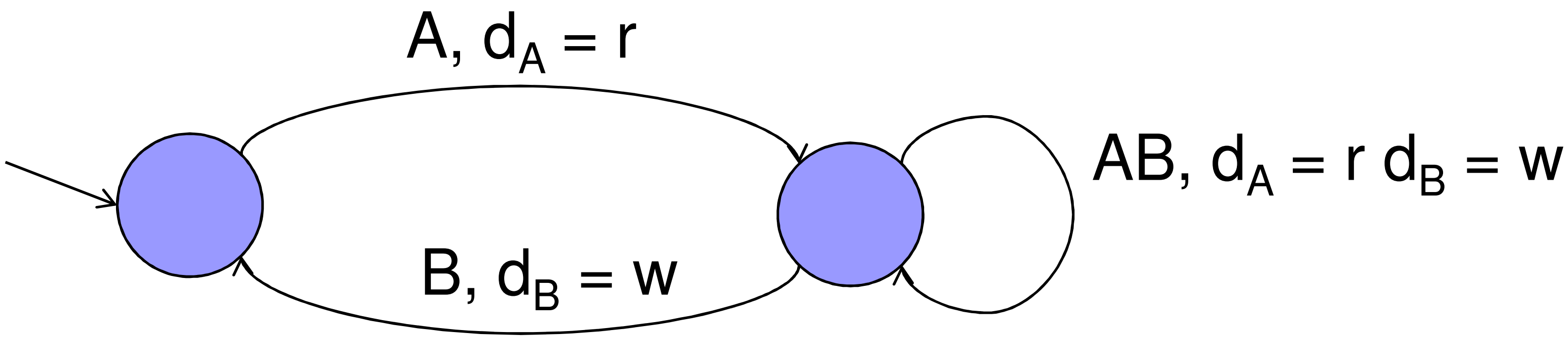
CA acceptance



- CA acceptance condition is implicitly fair
  - (A or) B cannot occur eventually always

A	r	r	r	r	r	r	r	r	r	r	…
	1										
B	w	w	w	w	w	w	w	w	w	w	…
	1.2	1.5	4.2	4.4	5	5.2	5.4	6	6.8	8	

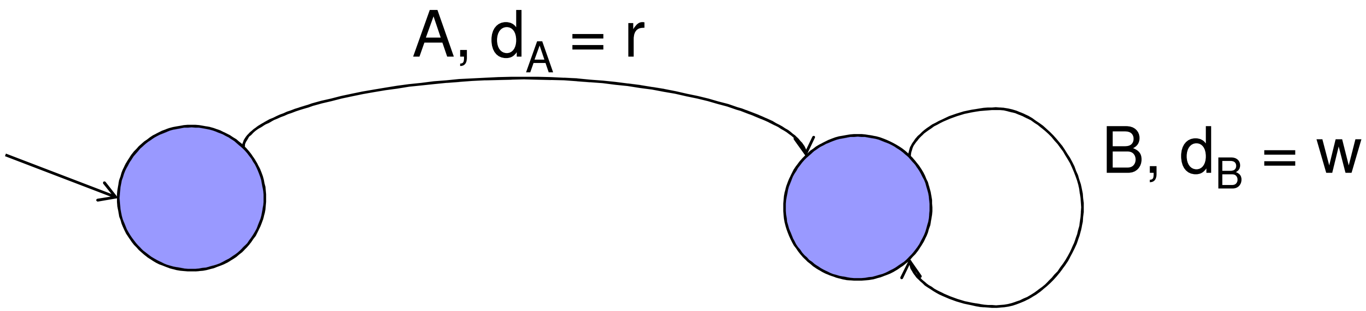
CA are fair, but not always …



- There exists accepting TDS where A and B never occur together

A	r	r	r	r	r	r	r	r	r	r	
	1	1.3	3	4.3	4.6	5.1	5.3	5.7	6.3	7	…
B	w	w	w	w	w	w	w	w	w	w	
	1.2	1.5	4.2	4.4	5	5.2	5.4	6	6.8	8	…

Which TDS is accepted?



- None, because
  - $A.\tau(1) > B.\tau(k)$, $\lim_K B.\tau(k) = \infty$ and $A.\tau(k) < \infty$

A	r	v	x	w	y	z	v	v	z	x	…
	1										
B	w	w	w	w	w	w	w	w	w	w	
	1.2	1.5	4.2	4.3	5	5.1	5.2	6	6.8	8	…

Overview

- Past [2002-2007]
- Present [end 2008 - mid 2009]
- Future [june 2009 -]

TDS vs streams of records

- Forget time and use infinite sequences

A

v	w	x	…
1	3	4	

B

v	w	…
2	4	

A=v	B=v	A=w	A=x,B=w	…

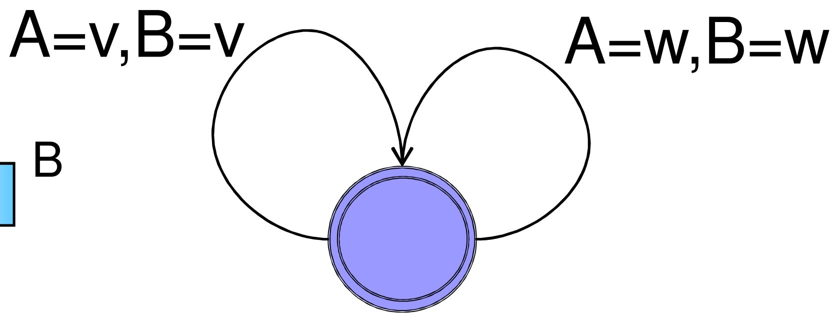
Büchi automata

- Extension of finite state automata

- A Büchi automaton accepts an infinite sequence (stream) if there exists a run of the automaton which visits at least one of the final states infinitely often.

Büchi automata for Reo

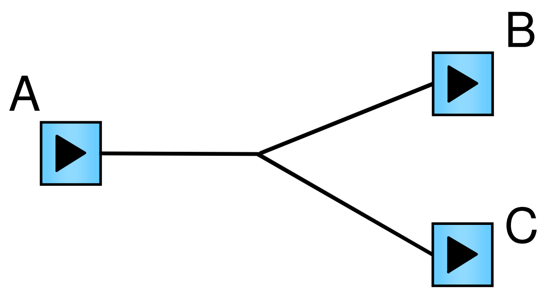
- Sync

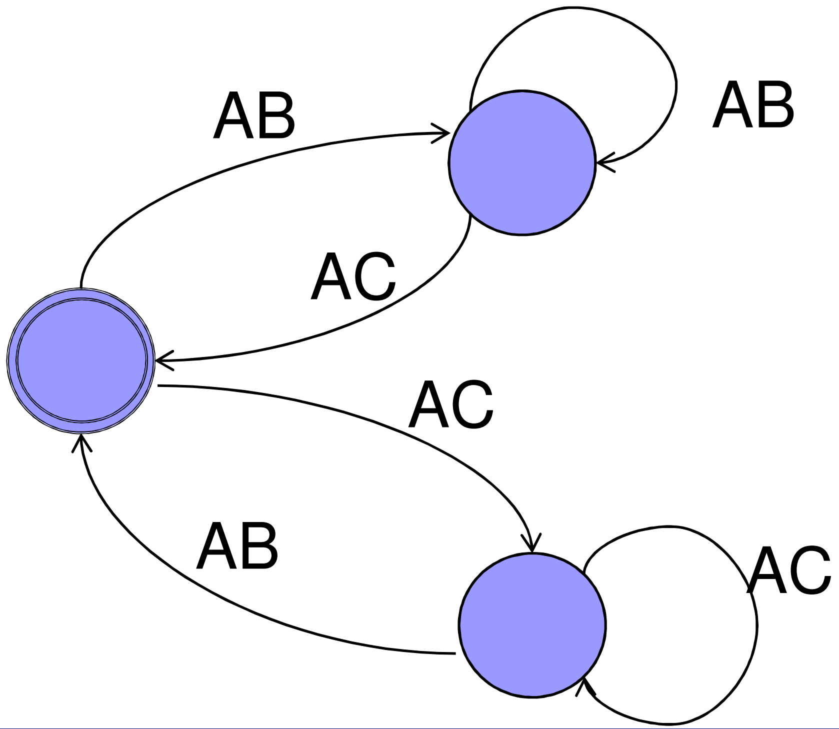


- If time in TDS is allowed to be ∞ then CA are essentially the same as BA with all states as final.

Fair connectors

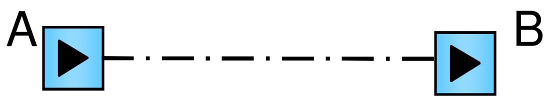
- FairMerger

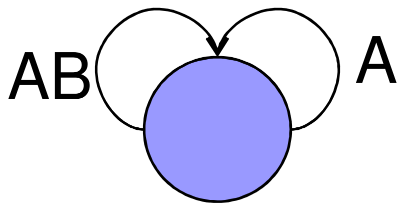




Context dependencies

- The behaviour can change depending upon presence and absence of I/O requests
- CA cannot model absence of I/O requests, thus context dependencies are reduced to (fair?) choices
  - Lossy synch





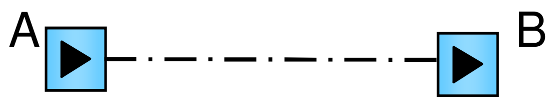
Guarded streams

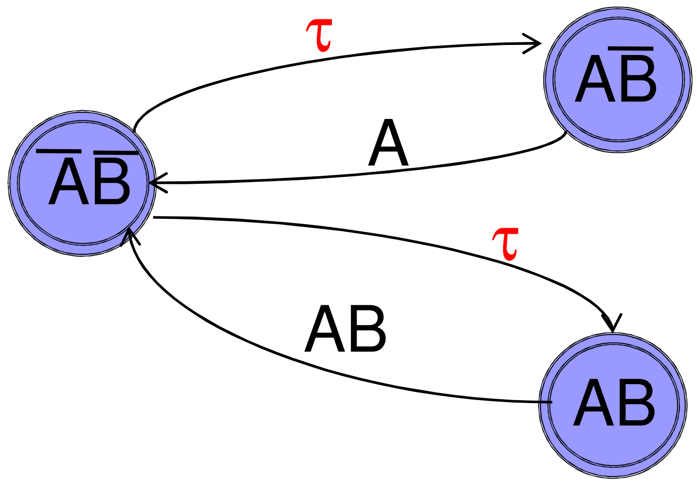
- Stream of pairs <r,f> where
  - r is a valuation over the ports, i.e. the present and absent I/O requests
  - f is the set of firing ports

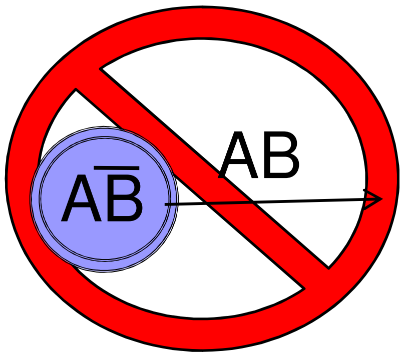
I/O request	AB	$A\overline{B}$	$\overline{A}\,\overline{B}$	AB	
firing	AB	A	$\varnothing$	A	…

Augmented Büchi Automata

- States are labeled by preconditions that must hold before taking an outgoing transition

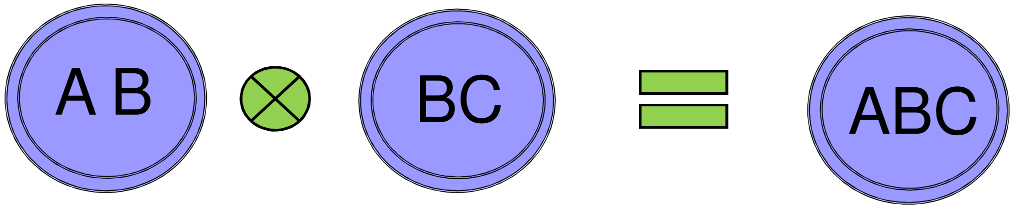




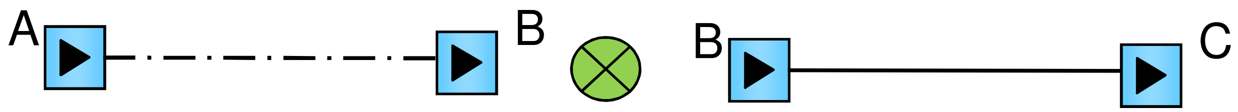


Composition

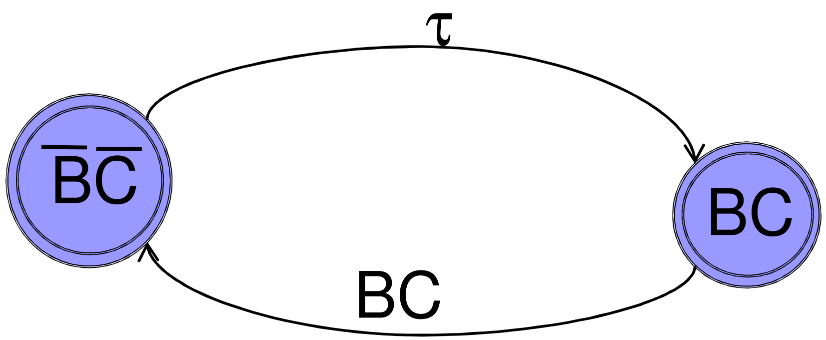
- Similar to CA, but
  - Final states as for Buchi automata
  - States labeled by the conjunction of the component labels



Context propagation



- Context propagation must be hard coded
  - Synchronous channel



Model Checking

- Action based LTL

$$\phi ::= T \mid \neg\phi \mid \phi \wedge \phi \mid r \mid \langle f \rangle \phi \mid \phi \, U \, \phi$$

requests

firings

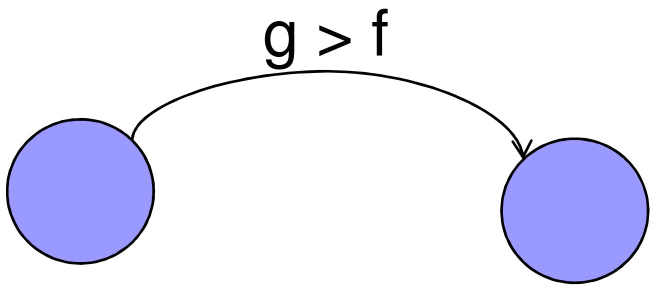
- More expressive than data stream logic
- On the fly model checking

Overview

- Past [2002-2007]
- Present [end 2008 - mid 2009]
- Future [june 2009 -]

Reo automata

- Transition system accepting guarded strings

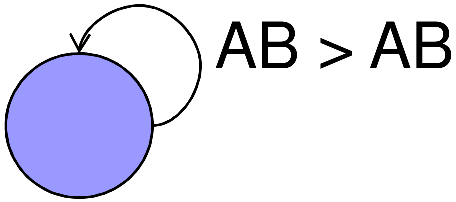


g = request guard
f = firing ports

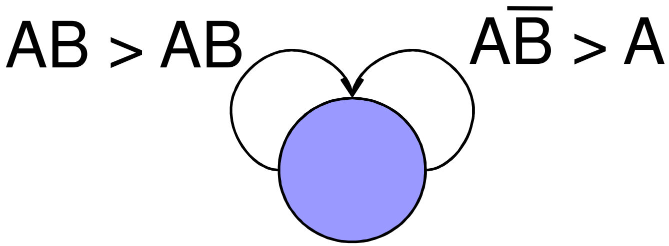
such that

- Observable = firing is not empty
- Reactive = data flow only where requests are made
- Uniform = removing unfired requests does not affect firing

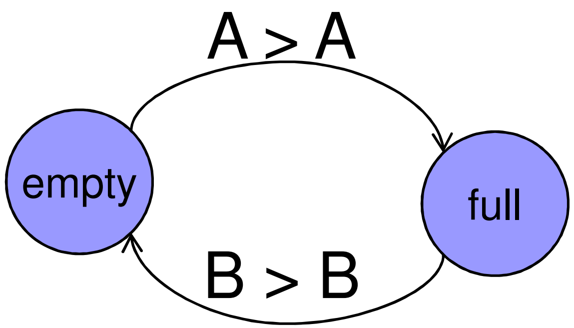
Reo automata



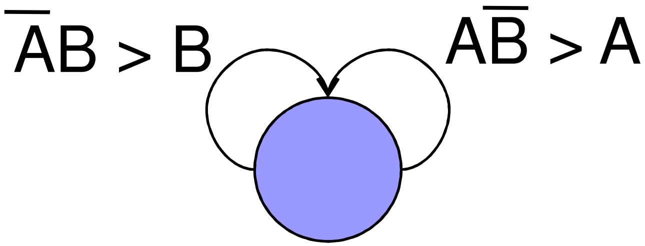
Sync



LossySync



FIFO1



AsyncDrain

Product

- Composition of two disjoint automata making transitions firing in parallel

$$(p,q) \xrightarrow{g \wedge g' > f \cup f'} (p',q')$$

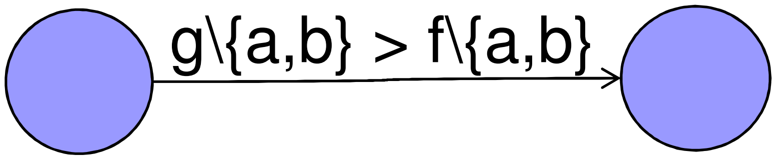
and in interleaving when one is not able to fire

$$(p,q) \xrightarrow{g \wedge q^{\#} > f} (p',q)$$

Here $q^{\#}$ is the negation of all guards outgoing from q.

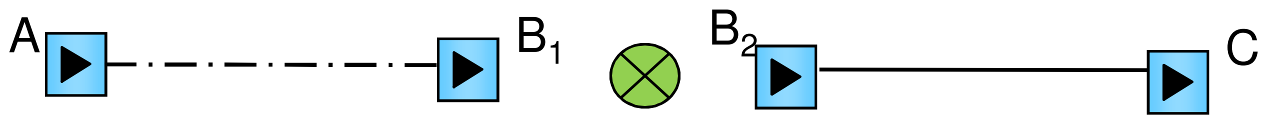
Synchronizing ports a and b

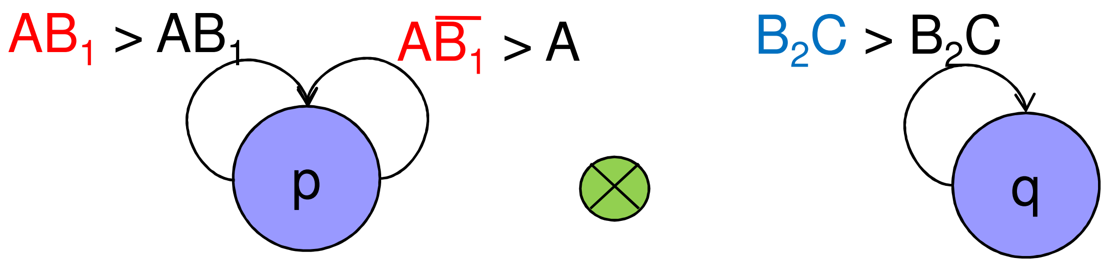
- Sub-automaton keeping only transitions



where

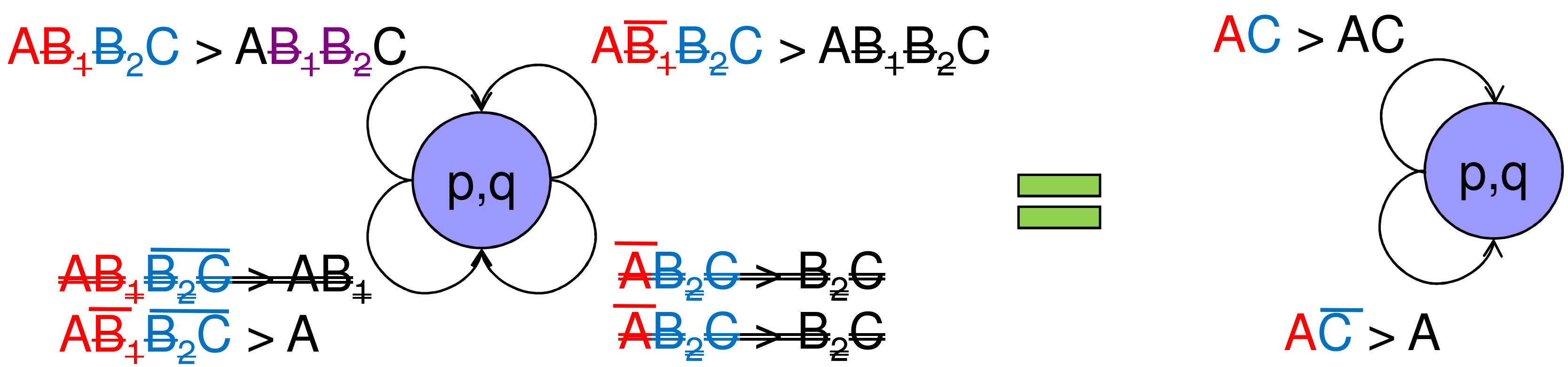
- both a and b are in firing set f (but are not alone)
- neither a nor b are in firing set f
- a or b are “present” in request g (self-pumping port)





LossySynch

Synch



Properties

- Sync is identity (up to renaming)
- Product is associative and commutative
- Synchronization is commutative and distribute with product

Final semantics

- Deterministic Reo automata with final states are coalgebra

$$Q \rightarrow 2 \times (1+Q)^{At_{\Sigma} \times 2^{\Sigma}}$$

- Final coalgebra = non empty and prefix closed subsets of $2^{At_{\Sigma} \times 2^{\Sigma}}$

- See tomorrow Alexandra's talk for specification language, synthesis, and equational logic.

Conclusions

- Constraint automata are fine but not with TDS semantics and not for context dependency.
- Buchi automata for Reo are good but somentimes not intuitive.
- Reo automata needs more investigation.

Shoot your questions …