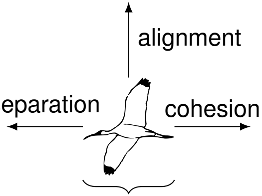
Imperial College
London

From Communicating Machines to Graphical Choreographies

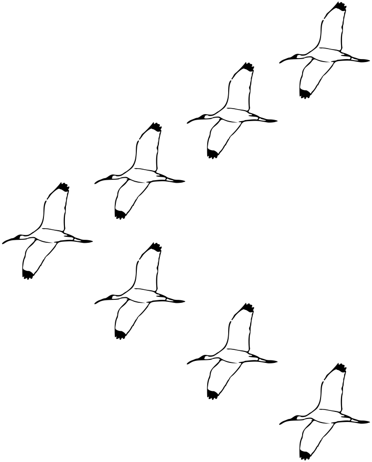
Julien Lange

based on joint works with L. Bocchi, E. Tuosto, *and* N. Yoshida

May 2015



Single bird $\simeq$ local behaviour $\simeq$ CFSM

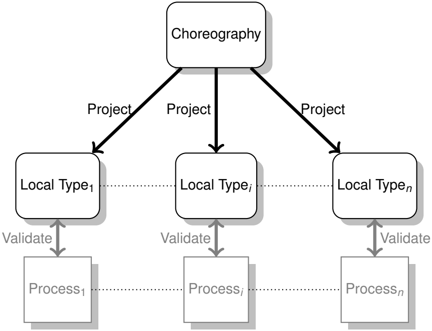


Flock $\simeq$ global behaviour $\simeq$ choreography

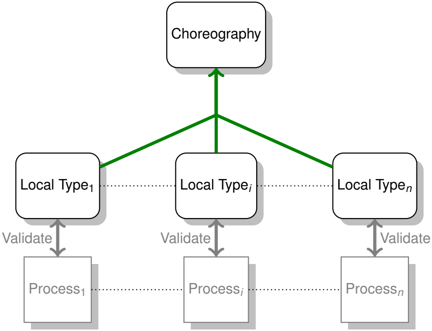
Introduction

- Parts of distributed systems change/evolve, not always in a coordinated way,
- these changes are often *not* documented.
- Service oriented systems are sometimes composed dynamically,
- it is often unclear how complex the overall system has become.
- Cognizant's Zero Deviation Lifecyle Business Unit.

A *global* point of view of a distributed system is *essential* for top-level management.

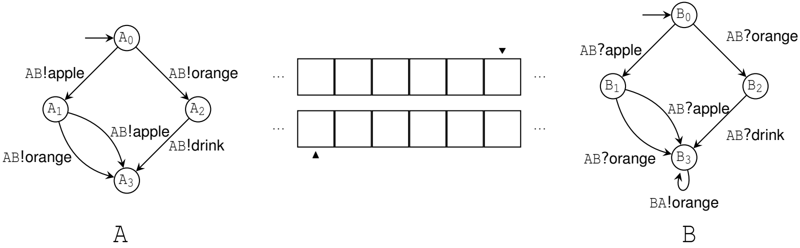


- Choreography-driven development, cf. Multiparty Session Types top-down approach (POPL'08 & ESOP'12)
- Not applicable without *a priori knowledge* of a choreography



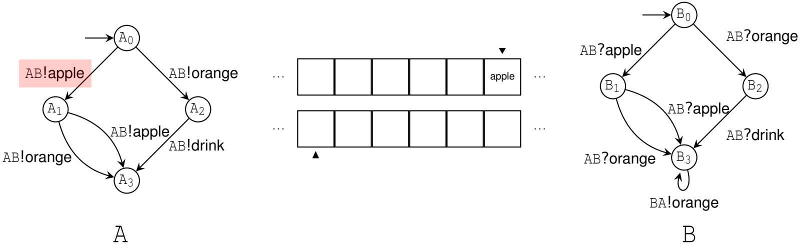
- Choreography-driven development, cf. Multiparty Session Types top-down approach (POPL'08 & ESOP'12)
- Not applicable without *a priori knowledge* of a choreography
- Our goal: from *Communicating Finite-State Machines* to *Global Graphs*

Background: CFSMs



"On Communicating Finite-State Machines", Brand & Zafiropulo ('83)

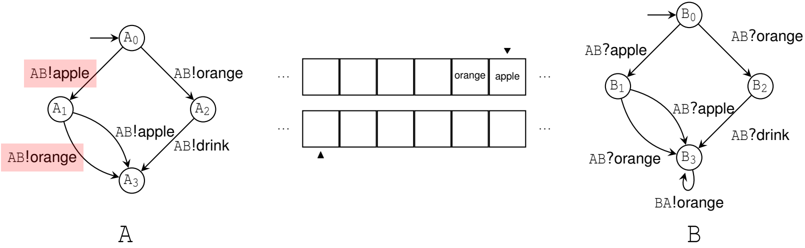
Background: CFSMs



"On Communicating Finite-State Machines", Brand & Zafiropulo ('83)

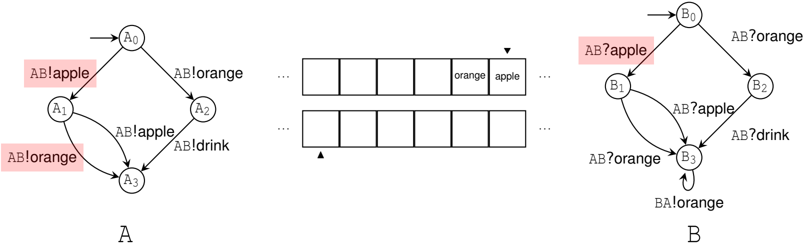
Imperial College London

Background: CFSMs



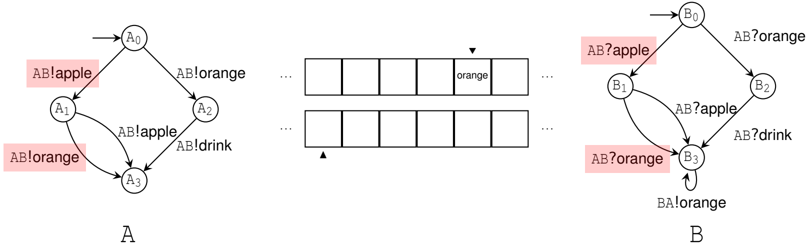
“On Communicating Finite-State Machines”, Brand & Zafiropulo (’83)

Background: CFSMs



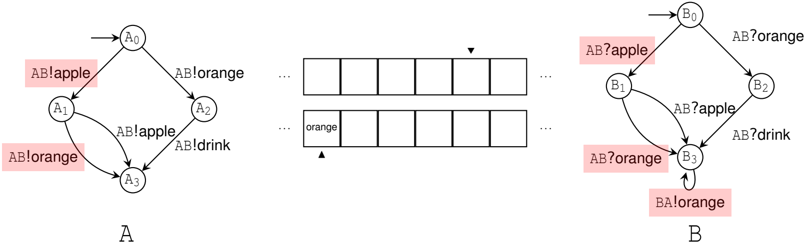
“On Communicating Finite-State Machines”, Brand & Zafiropulo (’83)

Background: CFSMs



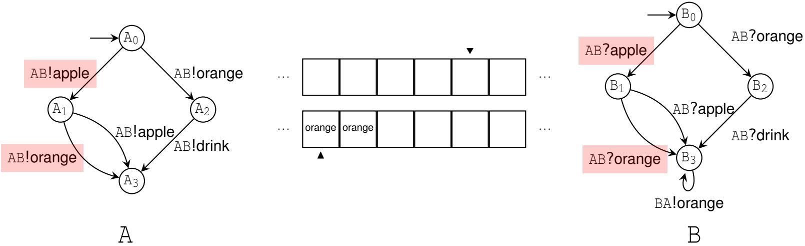
"On Communicating Finite-State Machines", Brand & Zafiropulo ('83)

Background: CFSMs



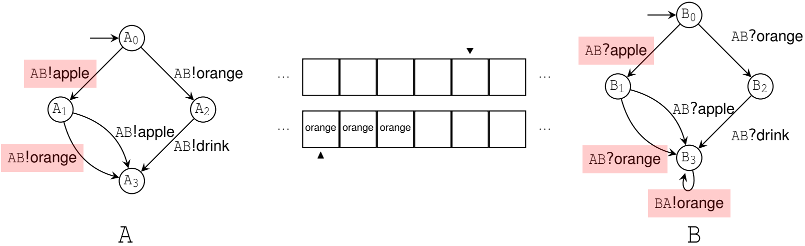
"On Communicating Finite-State Machines", Brand & Zafiropulo ('83)

Background: CFSMs



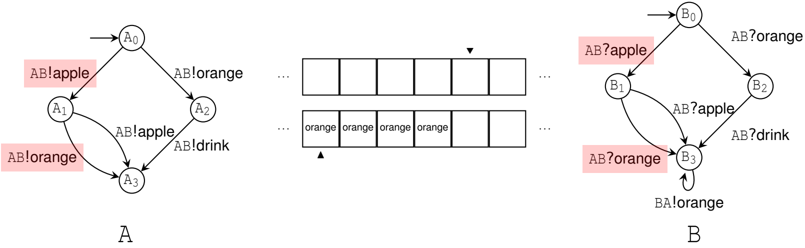
“On Communicating Finite-State Machines”, Brand & Zafiropulo (’83)

Background: CFSMs



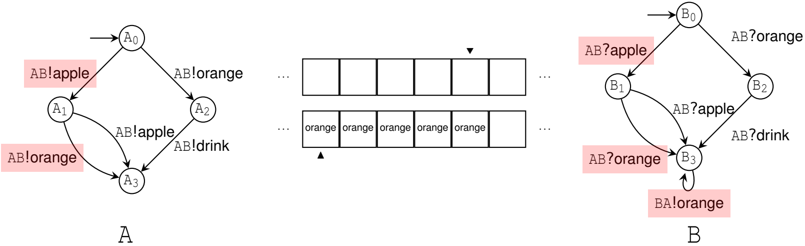
"On Communicating Finite-State Machines", Brand & Zafiropulo ('83)

Background: CFSMs



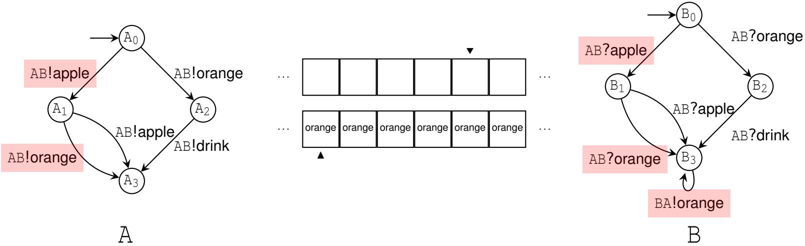
"On Communicating Finite-State Machines", Brand & Zafiropulo ('83)

Background: CFSMs



"On Communicating Finite-State Machines", Brand & Zafiropulo ('83)

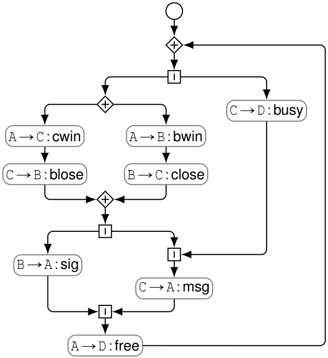
Background: CFSMs



"On Communicating Finite-State Machines", Brand & Zafiropulo ('83)

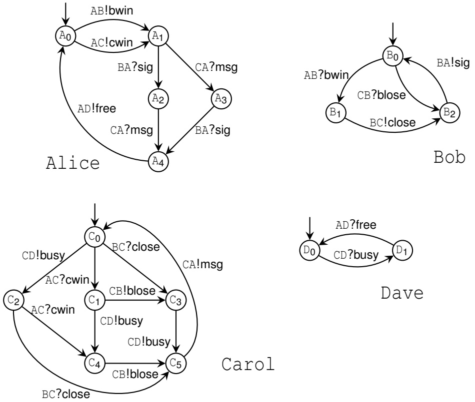
Imperial College London

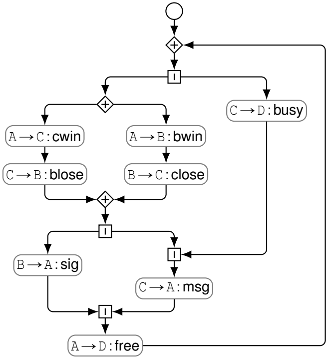
Global Graphs



Four Player Game

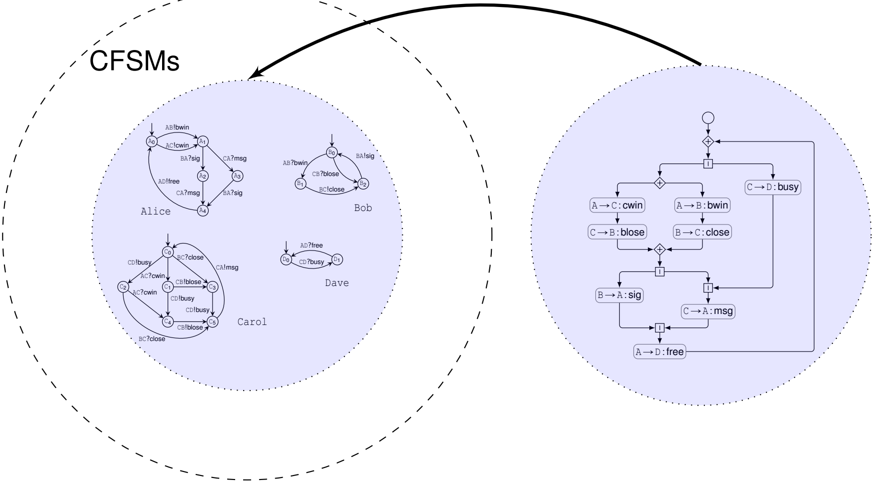
Global Graphs



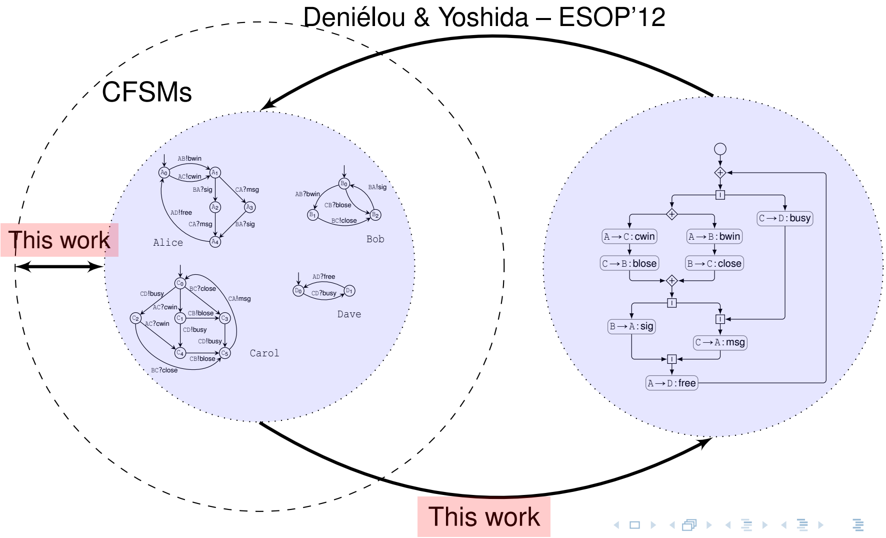


Four Player Game

Deniélou & Yoshida – ESOP'12



Imperial College
London



Objectives

Two main objectives:

- **Sound Condition for Safety:** generalised multiparty compatibility

  If $S = (M_1, \ldots, M_k)$ is *compatible* then S is "safe", i.e., every sent message is eventually received and no deadlock.

- **Construction of a Global Graph:**

  If G is the global graph constructed from S, then

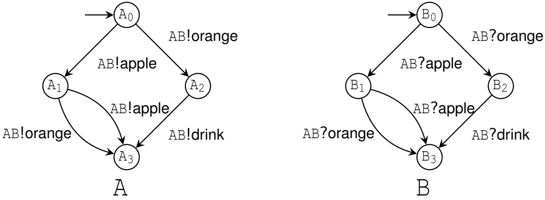
$$S = (M_1, \ldots, M_k) \equiv (G\downharpoonleft_1, \ldots, G\downharpoonleft_k)$$

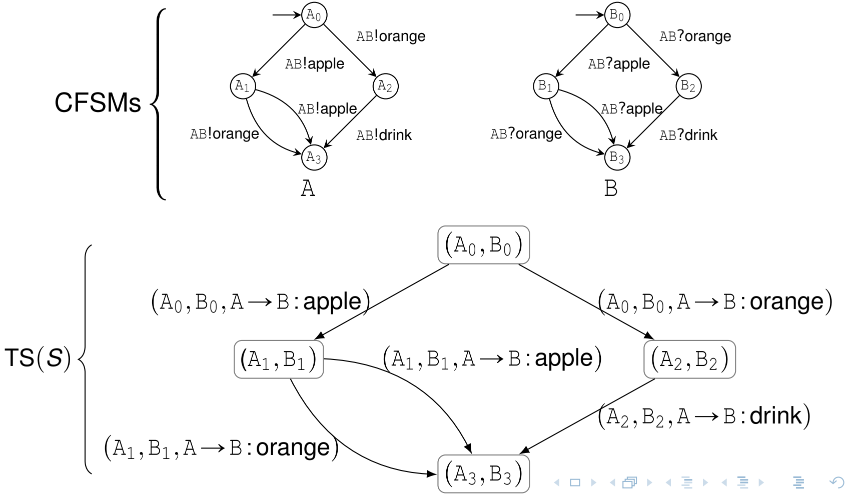
The Plan

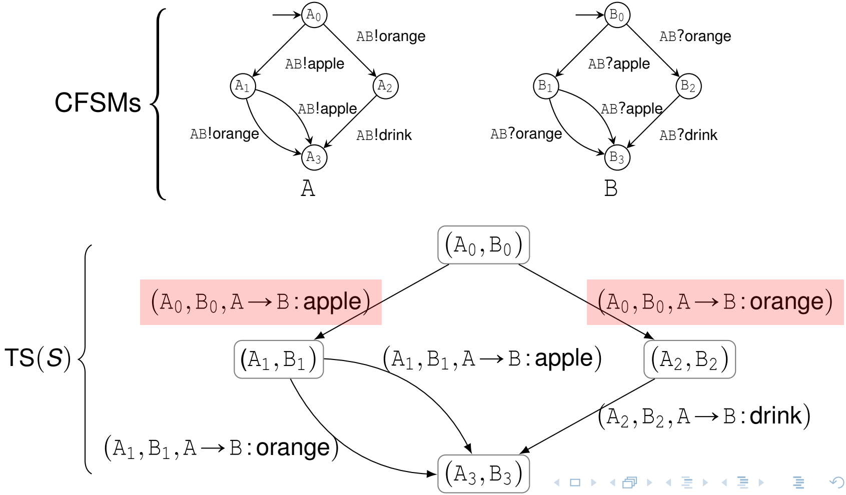
1. Build $\mathsf{TS}(S)$, the transition system of all *synchronous* executions
2. Check for safety on $\mathsf{TS}(S)$ to
   - ensure equivalence between original system and the projections of the choreography,
   - guarantee safety (no deadlock, no orphan message)
3. Build a choreography (global graph) from $\mathsf{TS}(S)$, relying on
   - the theory of regions, and
   - safe Petri nets.

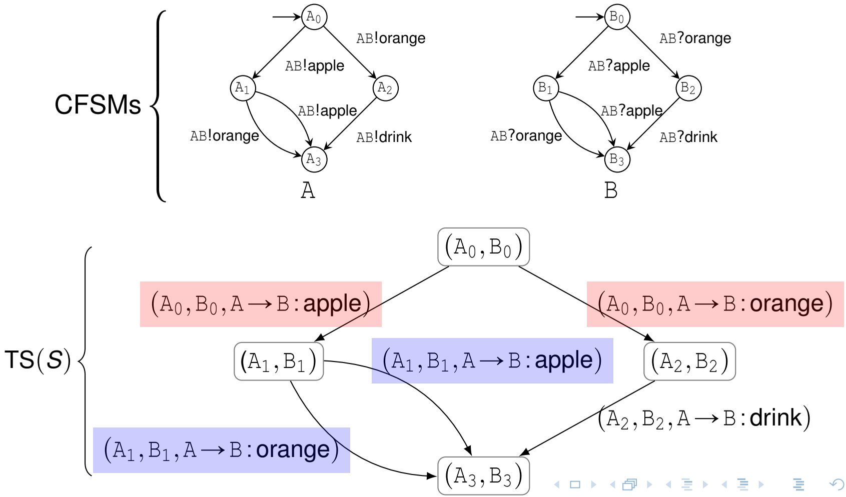
1. Synchronous Transition System of CFSMs

CFSMs

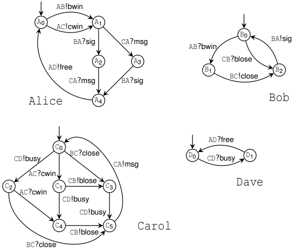

A₀
AB!orange
AB!apple
A₁
A₂
AB!apple
AB!orange
AB!drink
A₃
A
B₀
AB?orange
AB?apple
B₁
B₂
AB?apple
AB?orange
AB?drink
B₃
B


CFSMs
A0
AB!orange
AB!apple
A1
A2
AB!apple
AB!orange
AB!drink
A3
A
B0
AB?orange
AB?apple
B1
B2
AB?apple
AB?orange
AB?drink
B3
B
TS(S)
(A0, B0)
(A0, B0, A → B : apple)
(A0, B0, A → B : orange)
(A1, B1)
(A1, B1, A → B : apple)
(A2, B2)
(A2, B2, A → B : drink)
(A1, B1, A → B : orange)
(A3, B3)

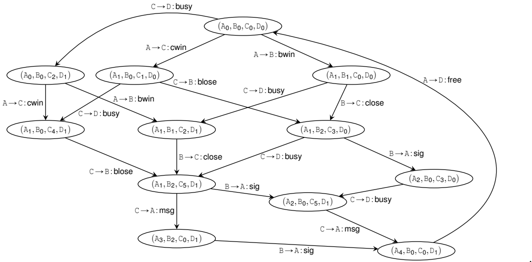

CFSMs
A0
AB!orange
AB!apple
A1
A2
AB!apple
AB!orange
AB!drink
A3
A
B0
AB?orange
AB?apple
B1
B2
AB?apple
AB?orange
AB?drink
B3
B
TS(S)
(A0, B0)
(A0, B0, A → B : apple)
(A0, B0, A → B : orange)
(A1, B1)
(A1, B1, A → B : apple)
(A2, B2)
(A2, B2, A → B : drink)
(A1, B1, A → B : orange)
(A3, B3)


CFSMs
A0
A1
A2
A3
AB!orange
AB!apple
AB!apple
AB!orange
AB!drink
A
B0
B1
B2
B3
AB?orange
AB?apple
AB?apple
AB?orange
AB?drink
B
TS(S)
(A0, B0)
(A0, B0, A → B : apple)
(A0, B0, A → B : orange)
(A1, B1)
(A1, B1, A → B : apple)
(A2, B2)
(A2, B2, A → B : drink)
(A1, B1, A → B : orange)
(A3, B3)

Synchronous Transition System (TS(S))



Four Player Game (CFSMs)



Synchronous Transition System (TS(S))

Imperial College London

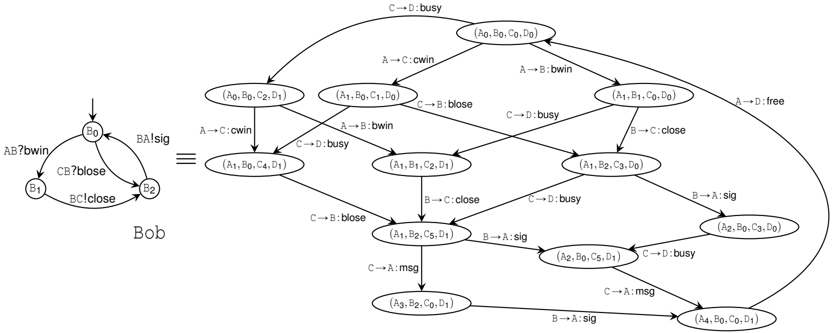
2. Check for Safety: *Generalised Multiparty Compatibility* (GMC)

1. Representability
2. Branching Property

Checking Compatibility (i) – Representability

Representability

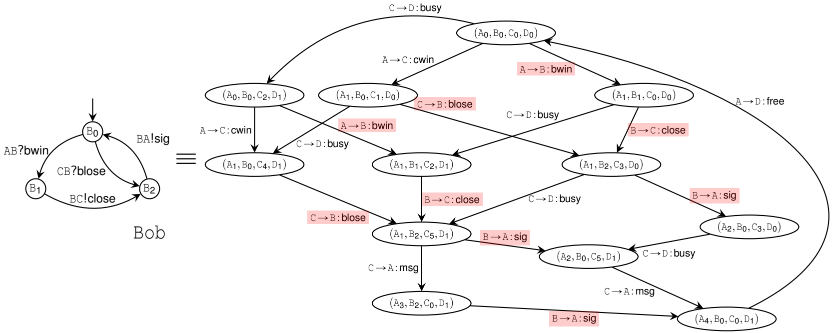
- The projected TS(S) $\equiv$ original machine
- Each branching in each machine must be represented in TS



Checking Compatibility (i) – Representability

Representability

- The projected TS$(S) \equiv$ original machine
- Each branching in each machine must be represented in TS

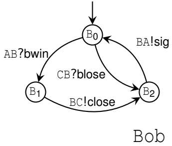


Imperial College London

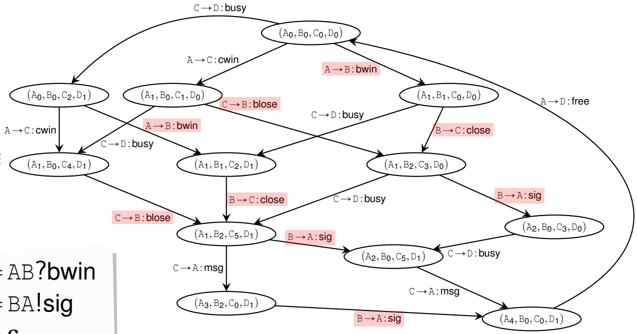
Checking Compatibility (i) – Representability

Representability

- The projected TS(S) $\equiv$ original machine
- Each branching in each machine must be represented in TS



$\equiv$

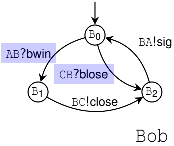


$$(\mathtt{A} \to \mathtt{B} : \mathsf{bwin}) \downharpoonright_{\mathtt{B}} = \mathtt{AB}?\mathsf{bwin}$$
$$(\mathtt{B} \to \mathtt{A} : \mathsf{sig}) \downharpoonright_{\mathtt{B}} = \mathtt{BA}!\mathsf{sig}$$
$$(\mathtt{C} \to \mathtt{D} : \mathsf{busy}) \downharpoonright_{\mathtt{B}} = \varepsilon$$

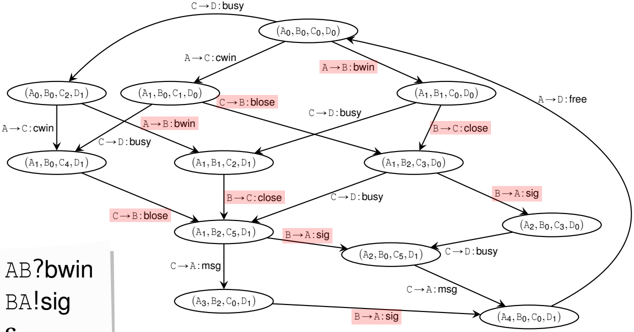
Checking Compatibility (i) – Representability

Representability

- The projected TS(S) $\equiv$ original machine
- Each branching in each machine must be represented in TS



$\equiv$

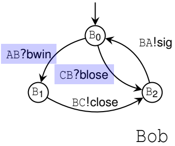


$$(\mathtt{A} \to \mathtt{B} : \mathsf{bwin}) \downharpoonright_{\mathtt{B}} = \mathtt{AB}?\mathsf{bwin}$$
$$(\mathtt{B} \to \mathtt{A} : \mathsf{sig}) \downharpoonright_{\mathtt{B}} = \mathtt{BA}!\mathsf{sig}$$
$$(\mathtt{C} \to \mathtt{D} : \mathsf{busy}) \downharpoonright_{\mathtt{B}} = \varepsilon$$

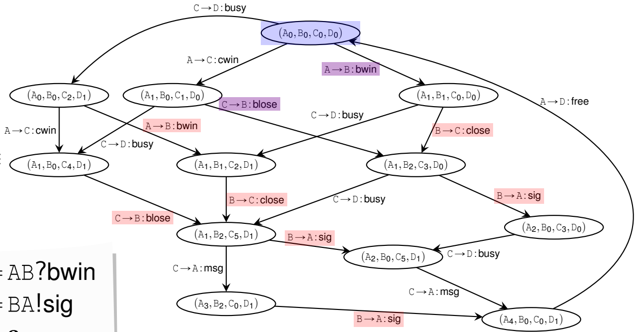
Checking Compatibility (i) – Representability

Representability

- The projected $\mathrm{TS}(S) \equiv$ original machine
- Each branching in each machine must be represented in TS



$\equiv$



$$(\mathtt{A} \to \mathtt{B} : \mathsf{bwin}) \downharpoonright_{\mathtt{B}} = \mathtt{AB}?\mathsf{bwin}$$
$$(\mathtt{B} \to \mathtt{A} : \mathsf{sig}) \downharpoonright_{\mathtt{B}} = \mathtt{BA}!\mathsf{sig}$$
$$(\mathtt{C} \to \mathtt{D} : \mathsf{busy}) \downharpoonright_{\mathtt{B}} = \varepsilon$$

Imperial College
London

Checking Compatibility (ii) – Branching Property

Each branching
$$\begin{tikzcd}[column sep=small] & n_1 \arrow[dl, "e"'] \arrow[dr, "e'"] & \\ {} & & {} \end{tikzcd}$$
in TS must be either

Checking Compatibility (ii) – Branching Property

Each branching $\begin{tikzcd} & n_1 \arrow[dl, "e"'] \arrow[dr, "e'"] & \\ {} & & {} \end{tikzcd}$ in TS must be either

- commuting: $$\begin{tikzcd} & n_1 \arrow[dl, "e"'] \arrow[dr, "e'"] & \\ n_2 \arrow[dr, "e'"'] & & n_3 \arrow[dl, "e"] \\ & n_4 & \end{tikzcd}$$

Checking Compatibility (ii) – Branching Property

Each branching
$$\begin{tikzcd} & n_1 \arrow[dl, "e"'] \arrow[dr, "e'"] & \\ {} & & {} \end{tikzcd}$$
in TS must be either

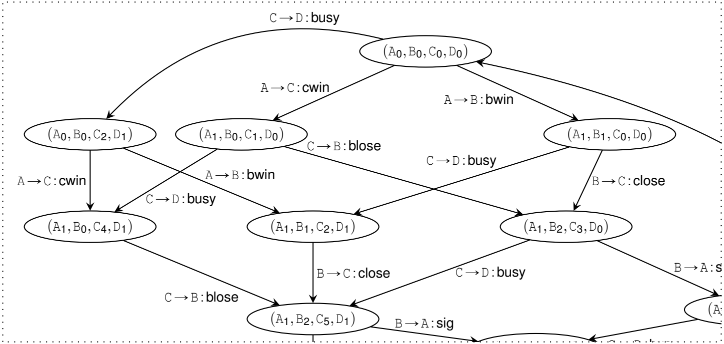
Last node, reachable from n_1, from which e and e' can be fired.

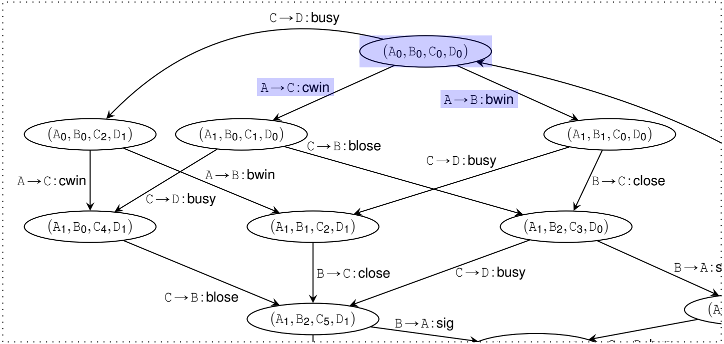
- or, each *last node* n_k

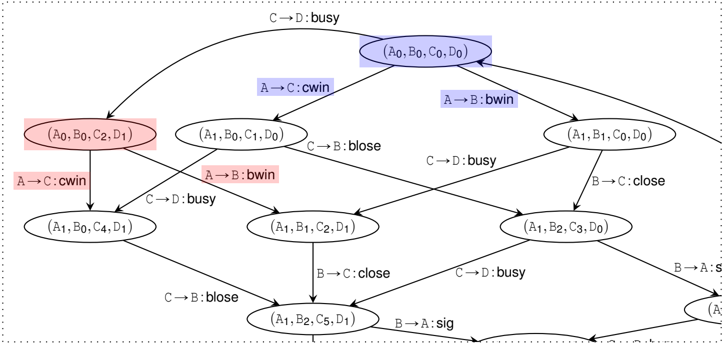
$$\begin{tikzcd} & n_1 \arrow[dl, "e"'] \arrow[dr, "e'"] \arrow[rr, "e_1"] & & n_2 \arrow[dl, "e"'] \arrow[dr, "e'"] \arrow[rr, dotted, "e_{k-1}"] & & \boxed{n_k} \arrow[dl, "e"'] \arrow[dr, "e'"] & \\ {} & & {} & & {} & & {} \end{tikzcd}$$

must be a "well-formed" choice, i.e.,

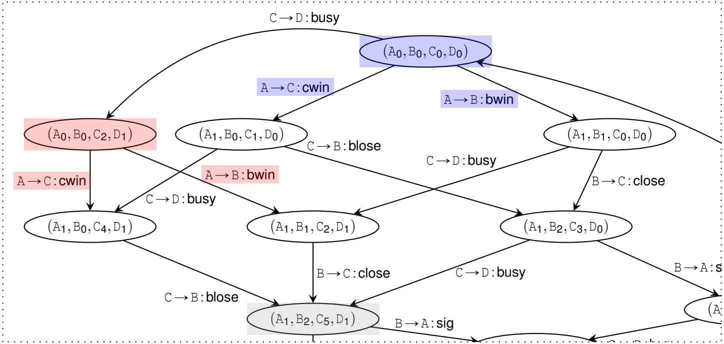
- for each participant
  - it receives a different message in each branch, or
  - it is not involved in the choice
- there is a unique sender
- there is no "race" between branches

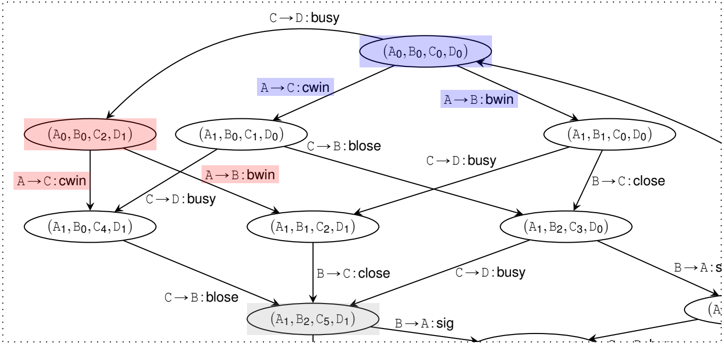

C → D : busy
(A0, B0, C0, D0)
A → C : cwin
A → B : bwin
(A0, B0, C2, D1)
(A1, B0, C1, D0)
C → B : blose
(A1, B1, C0, D0)
C → D : busy
A → C : cwin
A → B : bwin
B → C : close
C → D : busy
(A1, B0, C4, D1)
(A1, B1, C2, D1)
(A1, B2, C3, D0)
B → C : close
C → D : busy
B → A : s
C → B : blose
(A1, B2, C5, D1)
B → A : sig


C → D : busy
(A0, B0, C0, D0)
A → C : cwin
A → B : bwin
(A0, B0, C2, D1)
(A1, B0, C1, D0)
C → B : blose
(A1, B1, C0, D0)
C → D : busy
A → C : cwin
A → B : bwin
B → C : close
C → D : busy
(A1, B0, C4, D1)
(A1, B1, C2, D1)
(A1, B2, C3, D0)
B → C : close
C → D : busy
B → A : s
C → B : blose
(A1, B2, C5, D1)
B → A : sig


C → D : busy
(A0, B0, C0, D0)
A → C : cwin
A → B : bwin
(A0, B0, C2, D1)
(A1, B0, C1, D0)
C → B : blose
(A1, B1, C0, D0)
C → D : busy
A → C : cwin
A → B : bwin
B → C : close
C → D : busy
(A1, B0, C4, D1)
(A1, B1, C2, D1)
(A1, B2, C3, D0)
B → C : close
C → D : busy
C → B : blose
(A1, B2, C5, D1)
B → A : sig

Imperial College
London





$\mathtt{A}$: $\mathtt{AB}$!bwin $\neq$ $\mathtt{AC}$!cwin

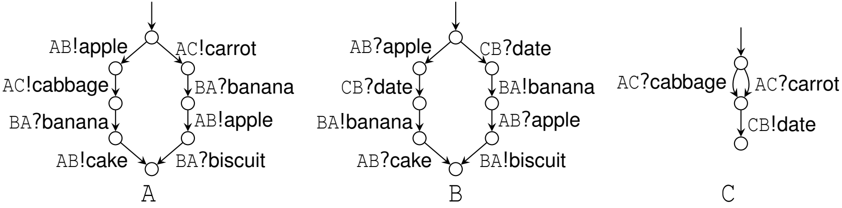
$\mathtt{B}$: $\mathtt{AB}$?bwin $\neq$ $\mathtt{CB}$?blose

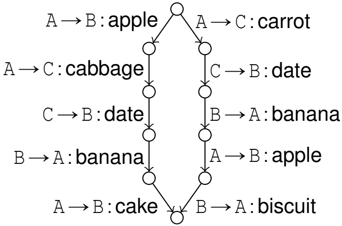
$\mathtt{D}$: not involved in the choice

$\mathtt{C}$: $\mathtt{AC}$?cwin $\neq$ $\mathtt{BC}$?close

No race, e.g., between $\mathtt{AC}$?cwin and $\mathtt{BC}$?close

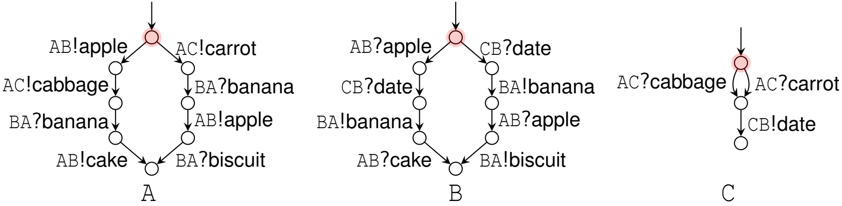
Violating the no-race condition

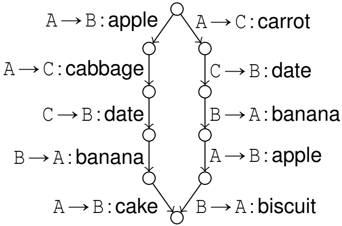




AB	:	ε
CB	:	ε
AC	:	ε
BA	:	ε

Violating the no-race condition

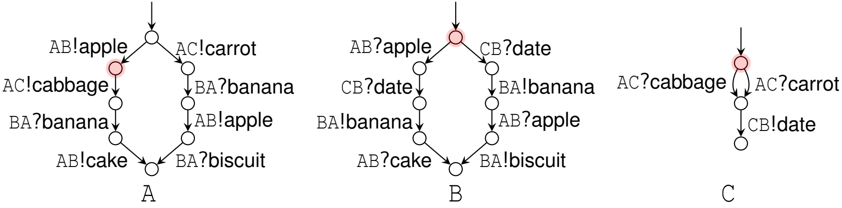


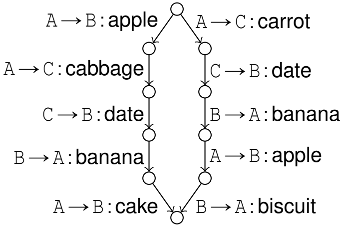


AB	:	ε
CB	:	ε
AC	:	ε
BA	:	ε

Imperial College
London

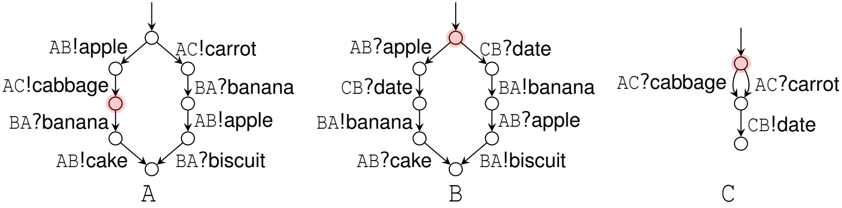
Violating the no-race condition

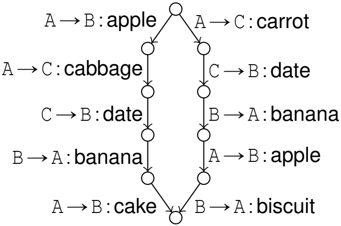




AB	:	apple
CB	:	ε
AC	:	ε
BA	:	ε

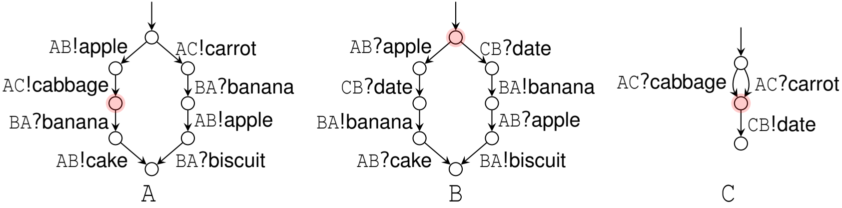
Violating the no-race condition

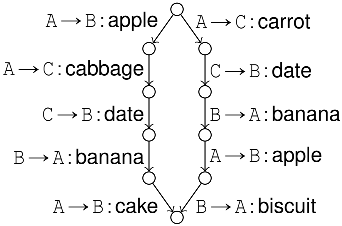




AB	:	apple
CB	:	ε
AC	:	cabbage
BA	:	ε

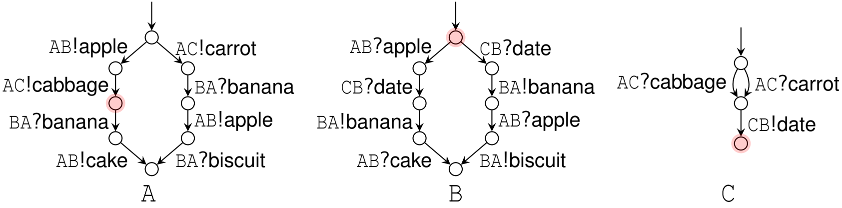
Violating the no-race condition

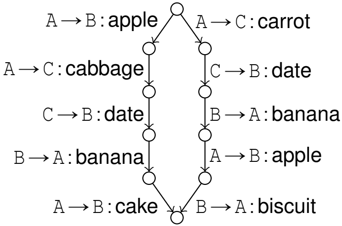




AB	:	apple
CB	:	ε
AC	:	ε
BA	:	ε

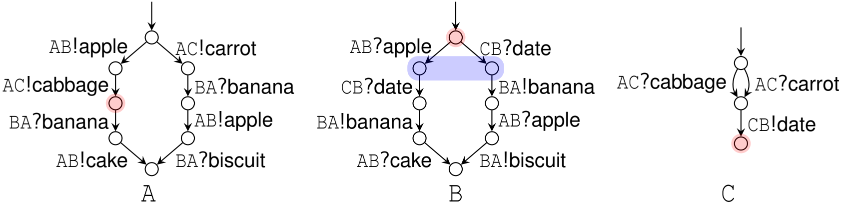
Violating the no-race condition

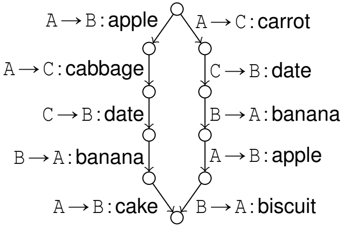




AB	:	apple
CB	:	date
AC	:	ε
BA	:	ε

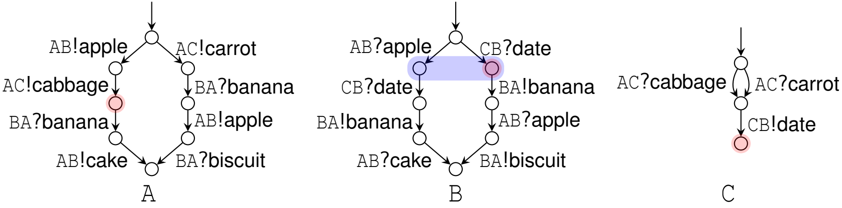
Violating the no-race condition

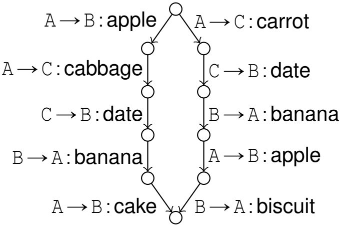




AB	:	apple
CB	:	date
AC	:	ε
BA	:	ε

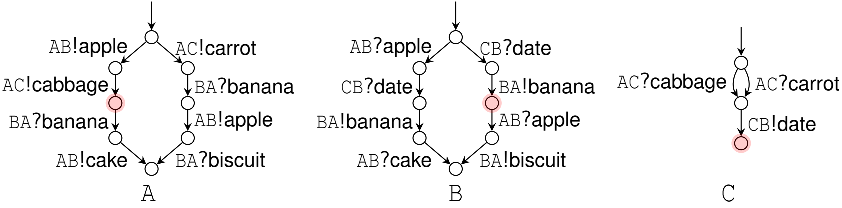
Violating the no-race condition

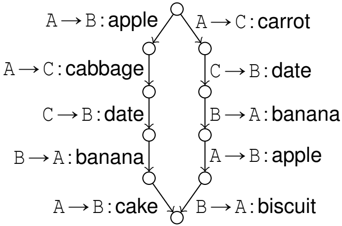




AB	:	apple
CB	:	ε
AC	:	ε
BA	:	ε

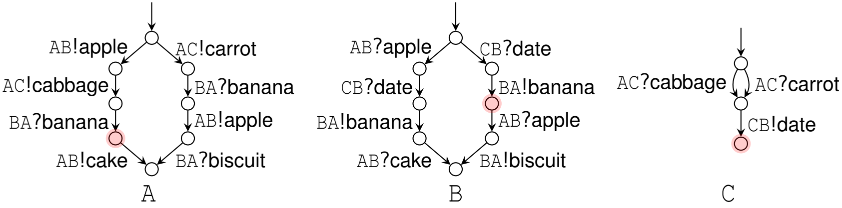
Violating the no-race condition

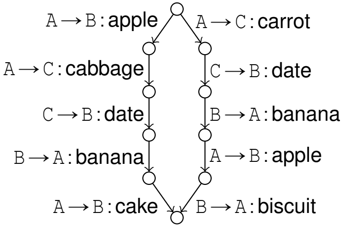




AB	:	apple
CB	:	ε
AC	:	ε
BA	:	banana

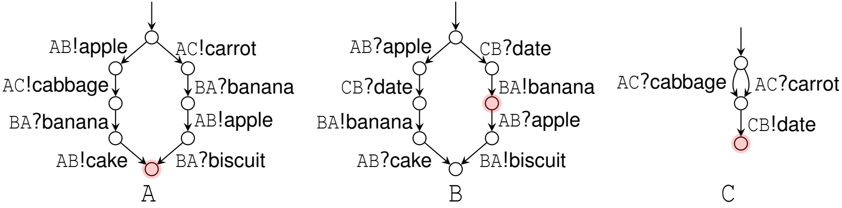
Violating the no-race condition

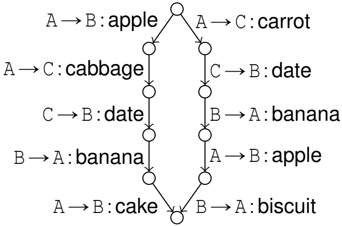




AB	:	apple
CB	:	ε
AC	:	ε
BA	:	ε

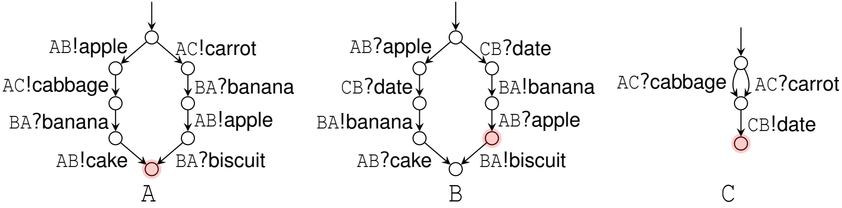
Violating the no-race condition

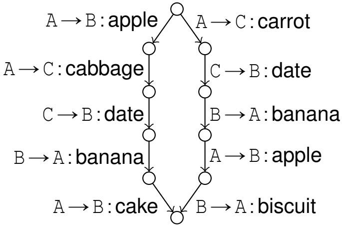




AB	:	apple · cake
CB	:	ε
AC	:	ε
BA	:	ε

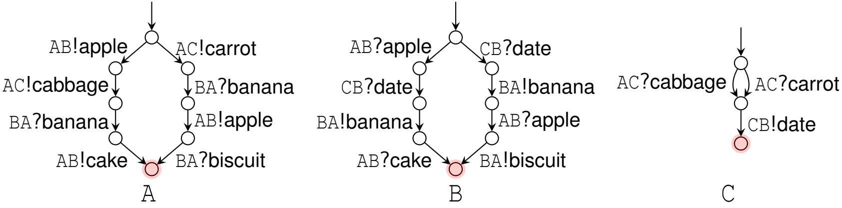
Violating the no-race condition

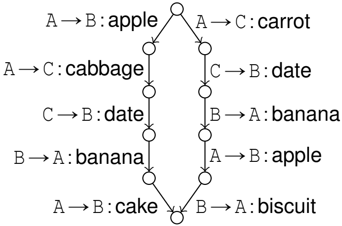




AB	:	cake
CB	:	ε
AC	:	ε
BA	:	ε

Violating the no-race condition

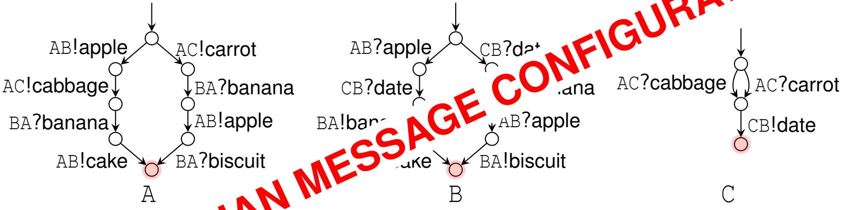


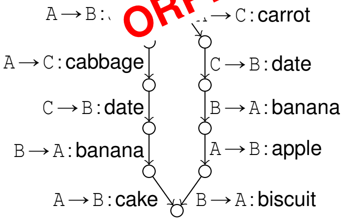


AB	:	cake
CB	:	ε
AC	:	ε
BA	:	biscuit

Violating the no-race condition

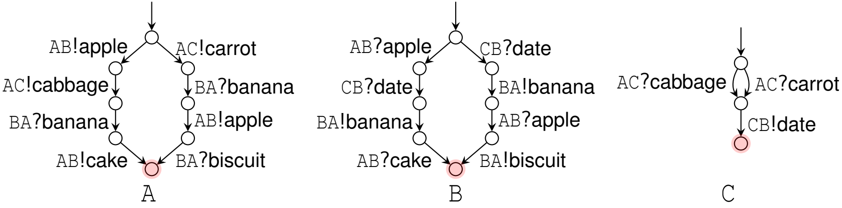
ORPHAN MESSAGE CONFIGURATION!

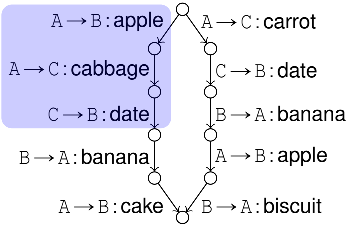




AB	:	cake
CB	:	ε
AC	:	ε
BA	:	biscuit

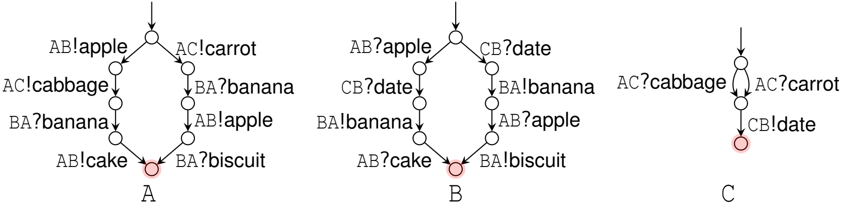
Violating the no-race condition

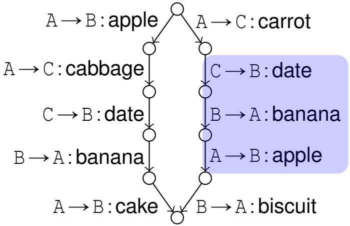




AB	:	cake
CB	:	ε
AC	:	ε
BA	:	biscuit

Violating the no-race condition



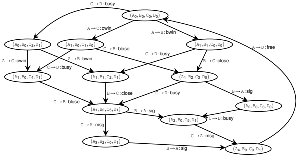


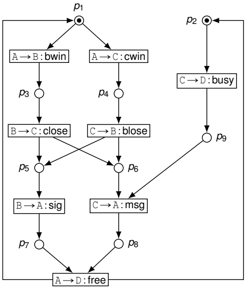
AB	:	cake
CB	:	ε
AC	:	ε
BA	:	biscuit

Imperial College London

3. Build a global graph

1: From TS to Petri Net

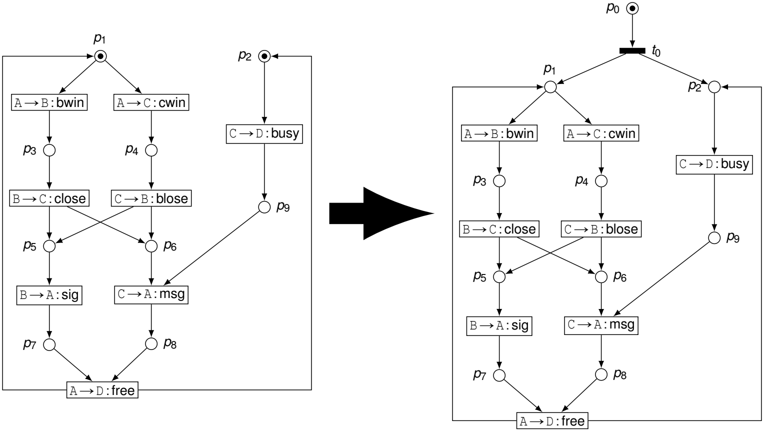




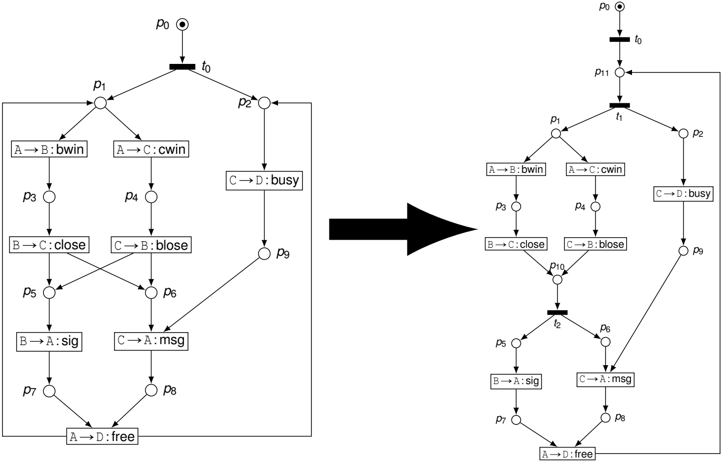
We use the work of Cortadella et al. (1998), based on the **theory of regions**, to derive a *safe* and *extended free-choice* Petri net from the Synchronous Transition System.

Imperial College London

2: From Petri Net to One-source Petri Net

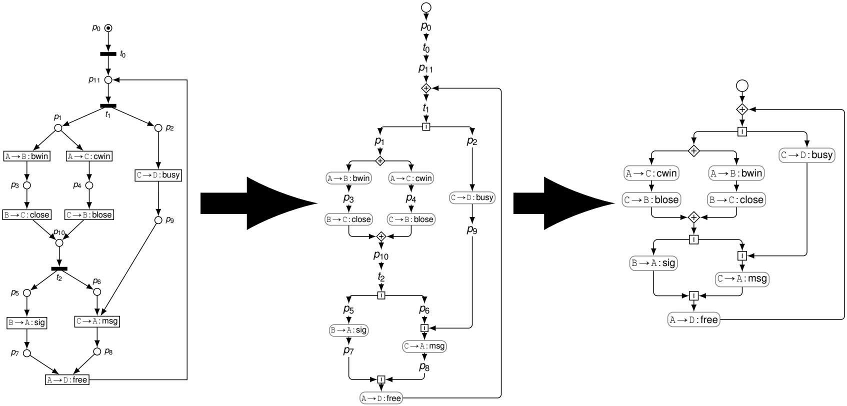


3: From One-source Petri Net to Joined Petri Net



Imperial College London

4 & 5 : From Joined Petri Net to Global Graph



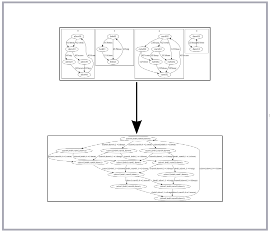
Prototype Implementation

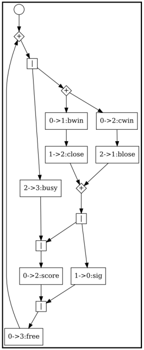
```
.outputs
.state graph
alice0 1 ! bwin alice1
alice0 2 ! cwin alice1
alice1 1 ? sig alice2
alice1 2 ! score alice3
alice2 2 ! score alice4
alice3 1 ? sig alice4
alice4 3 ! free alice0
.marking alice0
.end

.outputs
.state graph
bob0 0 ? bwin bob1
bob1 2 ! close bob2
bob0 2 ? blose bob2
bob2 0 ! sig bob0
.marking bob0
.end

.outputs
.state graph
carol0 1 ? close carol3
carol0 3 ! busy carol2
carol0 0 ? cwin carol1
carol1 3 ! busy carol4
carol1 1 ! blose carol3
carol2 1 ? close carol5
carol2 0 ? cwin carol4
carol3 3 ! busy carol5
carol4 1 ! blose carol5
carol5 0 ? score carol0
.marking carol0
.end

.outputs
.state graph
dave0 2 ? busy dave1
dave1 0 ? free dave0
.marking dave0
.end
```





https://bitbucket.org/julien-lange/gmc-synthesis

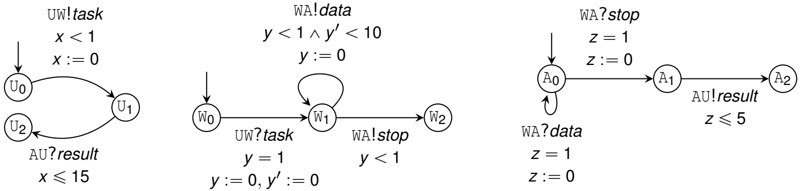
Experimental Evaluation

S	$\lvert\mathcal{P}\rvert$	$\lvert N\rvert$	$\lvert\Rightarrow\rvert$	GMC	$\lvert G\rvert$	Time (s)
Running Example	4	12	19	✓	16	0.184
Running Example $\times$ 2	8	144	456	✓	32	22.307
Bargain	3	4	4	✓	8	0.103
Bargain $\times$ 2	6	16	32	✓	16	0.161
Alternating 3-bit	2	24	48	✓	18	3.164
Alternating 3-bit $\times$ 2	4	576	2304	✓	34	12.069
TPMContract v2 $\times$ 2	4	25	80	✓	30	0.362
Sanitary Agency	4	17	21	✓	22	0.241
Sanitary Agency $\times$ 2	8	196	476	✓	44	3.165
Health System	6	10	11	✓	14	0.17
Health System $\times$ 2	12	100	220	✓	28	1.702
Logistic	4	13	17	✓	27	0.276
Logistic $\times$ 2	8	169	442	✓	54	2.155
Cloud System v4	4	7	8	✓	12	0.14
Cloud System v4 $\times$ 2	8	49	112	✓	24	0.432

Experimental Evaluation

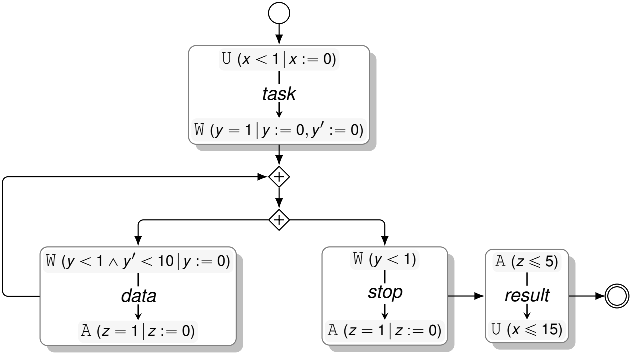
S	$\lvert\mathcal{P}\rvert$	$\lvert N\rvert$	$\lvert\Rightarrow\rvert$	GMC	$\lvert G\rvert$	Time (s)
Running Example	4	12	19	✓	16	0.184
Running Example × 2	8	144	456	✓	32	22.307
Bargain	3	4	4	✓	8	0.103
Bargain × 2	6	16	32	✓	16	0.161
Alternating 3-bit	2	24	48	✓	18	3.164
Alternating 3-bit × 2	4	576	2304	✓	34	12.069
TPMContract v2 × 2	4	25	80	✓	30	0.362
Sanitary Agency	4	17	21	✓	22	0.241
Sanitary Agency × 2	8	196	476	✓	44	3.165
Health System	6	10	11	✓	14	0.17
Health System × 2	12	100	220	✓	28	1.702
Logistic	4	13	17	✓	27	0.276
Logistic × 2	8	169	442	✓	54	2.155
Cloud System v4	4	7	8	✓	12	0.14
Cloud System v4 × 2	8	49	112	✓	24	0.432

From Communicating Timed Automata to Timed Choreographies



User (U) Worker (W) Aggregator (A)

- Each machine owns several clocks (not shared)
- Time elapses at the same pace for each clock



- Construction as in the untimed setting
- Additional safety checks for time constraints

Safety checks for CTAs

- MC: (Generalised) Multiparty Compatibility
- IE: Interaction Enabling
- CE: Cycle Enabling

Property	$S \sim (\mathrm{TS}(S)\downharpoonright_{\mathtt{p}})_{\mathtt{p} \in \mathcal{P}}$	Safety	Progress	Non-Zeno	Eventual Reception
MC	✓	✓	✗	✗	✗
MC+IE	✓	✓	✓	✗	✗
MC+CE	✓	✓	✗	✓	✗
MC+IE+CE	✓	✓	✓	✓	✓

L. Bocchi, J. Lange, and N. Yoshida. *Meeting Deadlines Together.*
Draft: `www.doc.ic.ac.uk/~jlange/cta/`

Summing up

- An effective way of checking properties of CFSMs, and whether one can construct a global graph from them.
- An algorithm based on the theory of regions.
- A CFSMs characterisation of well-formed *generalised global types*.
- `https://bitbucket.org/julien-lange/gmc-synthesis`
- An extension for communicating timed automata

References

- Julien Lange, Emilio Tuosto, and Nobuko Yoshida. “From Communicating Machines to Graphical Choreographies”. In: *POPL 2015*. 2015
- Laura Bocchi, Julien Lange, and Nobuko Yoshida. *Meeting Deadlines Together*. `www.doc.ic.ac.uk/~jlange/cta/`. 2015

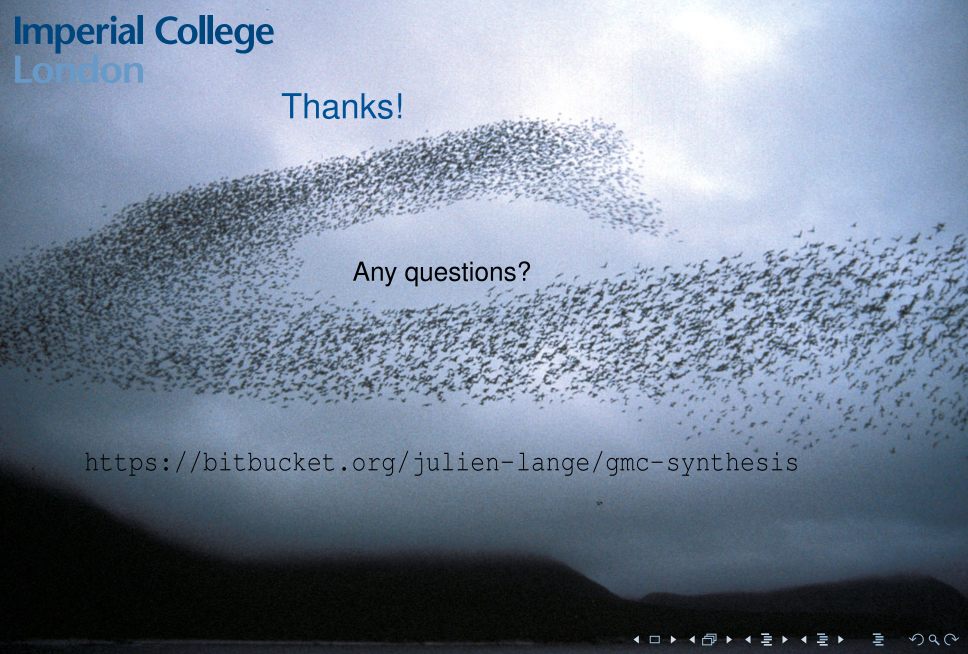
Related work (*i*)

- Pierre-Malo Deniélou and Nobuko Yoshida. “Multiparty Session Types Meet Communicating Automata”. In: *ESOP*. LNCS. Springer, 2012
- Julien Lange and Emilio Tuosto. “Synthesising Choreographies from Local Session Types”. In: *CONCUR*. LNCS. Springer, 2012
- Pierre-Malo Deniélou and Nobuko Yoshida. “Multiparty Compatibility in Communicating Automata: Characterisation and Synthesis of Global Session Types”. In: *ICALP*. LNCS. 2013
- Laura Bocchi, Weizhen Yang, and Nobuko Yoshida. “Timed Multiparty Session Types”. In: *CONCUR 2014*. 2014
- Pavel Krcal and Wang Yi. “Communicating Timed Automata: The More Synchronous, the More Difficult to Verify”. In: *CAV*. LNCS. 2006

Related work (*ii*)

- Kohei Honda, Nobuko Yoshida, and Marco Carbone. “Multiparty asynchronous session types”. In: *POPL*. ACM, 2008
- Dimitris Mostrous, Nobuko Yoshida, and Kohei Honda. “Global Principal Typing in Partially Commutative Asynchronous Sessions”. In: *ESOP*. LNCS. Springer, 2009
- Samik Basu, Tevfik Bultan, and Meriem Ouederni. “Synchronizability for Verification of Asynchronously Communicating Systems”. In: *VMCAI*. LNCS. Springer, 2012
- Samik Basu, Tevfik Bultan, and Meriem Ouederni. “Deciding choreography realizability”. In: *POPL*. ACM, 2012

Thanks!



Any questions?

`https://bitbucket.org/julien-lange/gmc-synthesis`