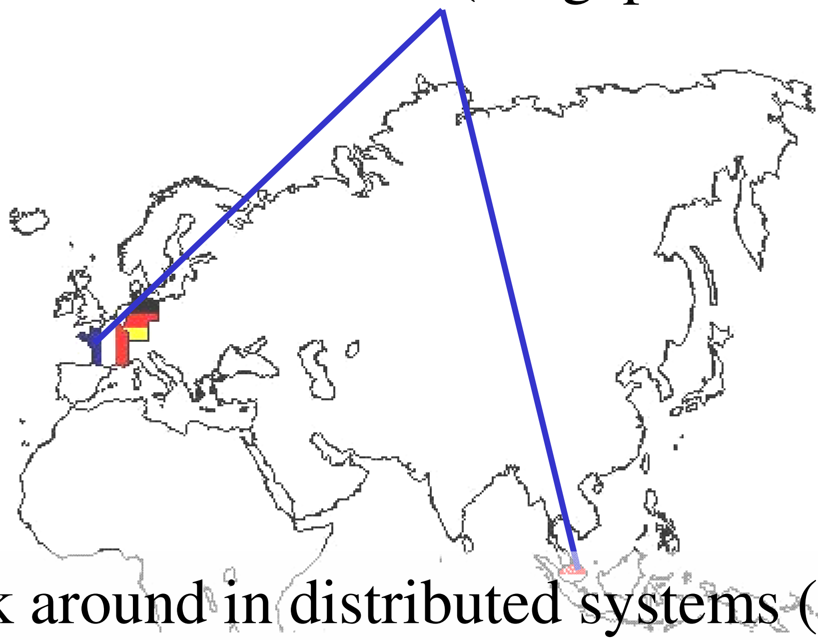
Realisability of
Message Sequence Charts

obtaining « easily » distributed implementation

Blaise Genest, CNRS (Singapore, Rennes)

IRISA Lab:
CNRS
Univ Rennes 1
INRIA

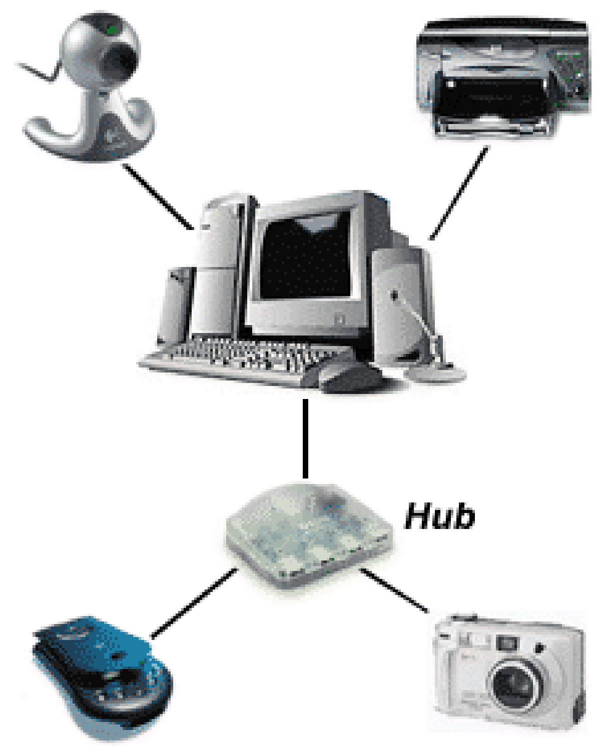


IPAL lab: CNRS+
NUS (Thiagarajan…),
A*STAR/I2R (R&D)

Work around in distributed systems (games, control, verif.)

INTRODUCTION

Why distributed implementation?



Because the world is distributed, communicant…

Communication protocol:

Control on 2 different processes (USB)

CardReader()
repeat
send(data) to pc

PC()
repeat
receive(data) from CardReader

Hard to write distributed implementation

Distributed programs	Human way of thinking
Globally parallel	Globally sequential

Difficult to write a distributed algorithm

→ Produce it automatically from "sequential" spec?

Q: Which model for sequential specification, for distributed algorithm?

Kind of models?

Most asyncrhonous system possible
(exit Petri Nets, Mazurkiewicz trace, product of automata since actions are blocking/synchronizing)

$\Rightarrow$Based on messages, with separeted send and receive.
$\Rightarrow$FIFO Channel between each pair of process
Can always send, can receive from a channel p only if non empty

Ex: telecomunication protocols etc.

Example of a specification hard to distribute

Specification:
2 Processes: 0, 1.
Both Processes can send a message to the other process.
After a message have been received, a new message can be sent.

But no message crossing.

Accept at any point when no message sent and not yet received.

Example of a specification hard to distribute

Specification:

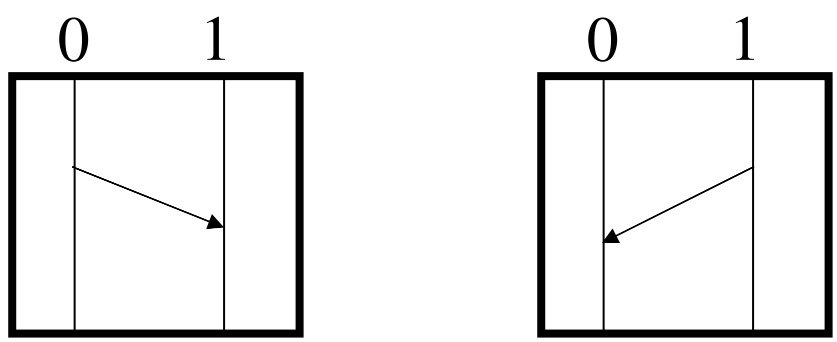
2 Processes: 0, 1.

Both Processes can send a message to the other process.

After a message have been received, a new message can be sent.

Accept at any point when no message sent and not yet received.

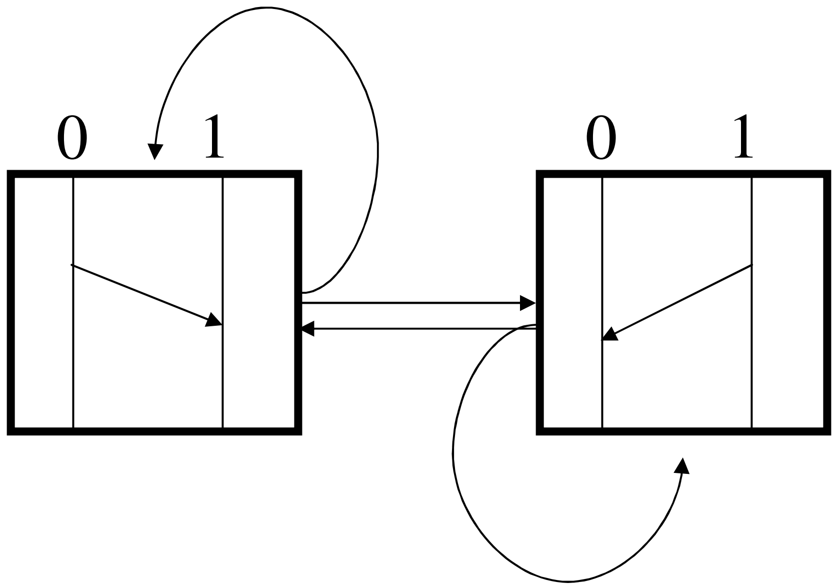
Q: How to modelize it with a computer science model?



2 scenarios

Example of a specification hard to distribute

Q: How to modelize it with a computer science model?

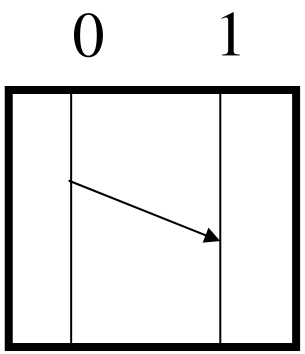


Graph of scenarios

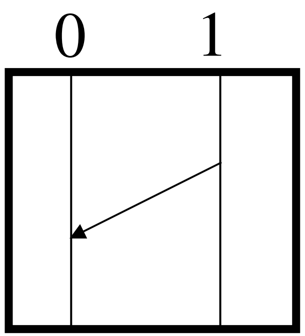
Example of a specification hard to distribute

Distributed Implementation:
Each process = Finite Automaton A_i with sends and receives

a =

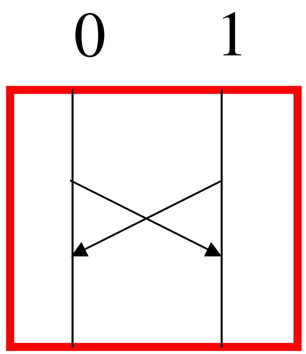


b =



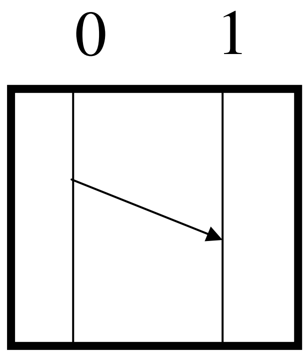
Accept if both process accepts

c =

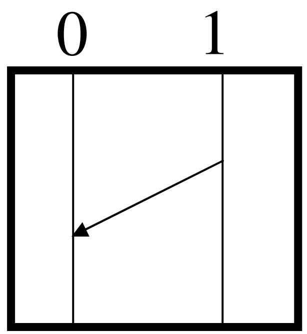


Example of a specification hard to distribute

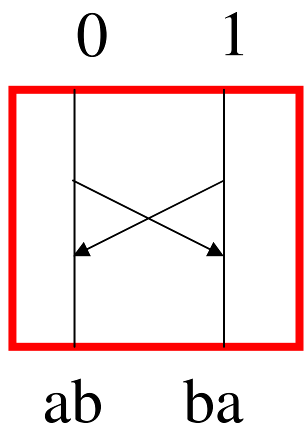
a =



b =



c =



(ab or ba) not possible to implement with
no information exchange between processes

c looks like something legitimate for each process.

Example of a specification hard to distribute

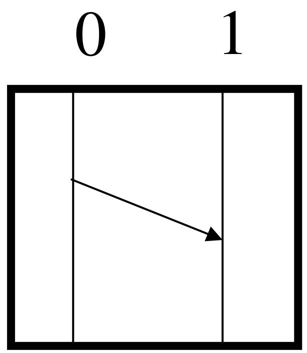
Distributed Implementation:
Each process = Finite Automaton A_i

different setting:
State of each process:

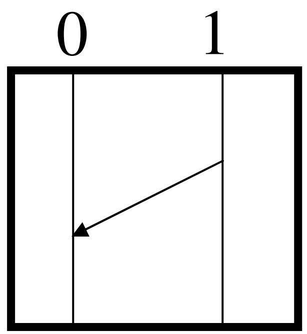
- attached to mesages sent by the process
- determined deterministically

Example of a specification hard to distribute

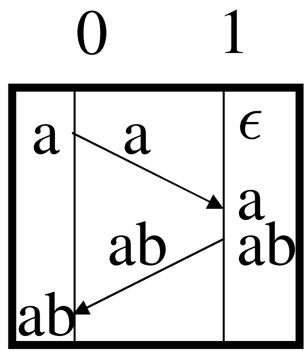
a =



b =



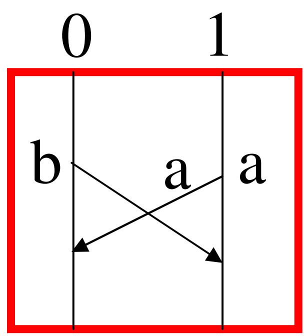
ab =



each process checks that
msg info extends local info

Each process remembers what it saw and tags messages with this memory

c =

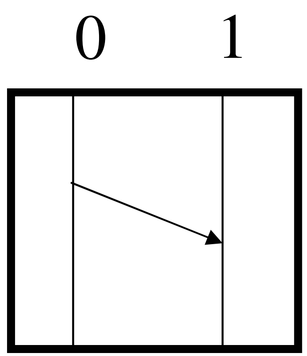


process 0 witness a problem and reject.

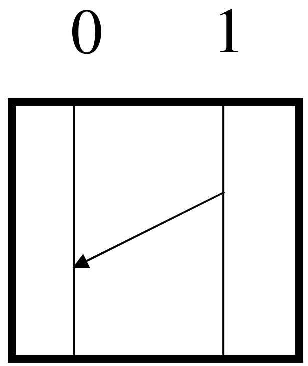
(ab or ba) easy to implement with deterministic additional information

Example of a specification hard to distribute

a =



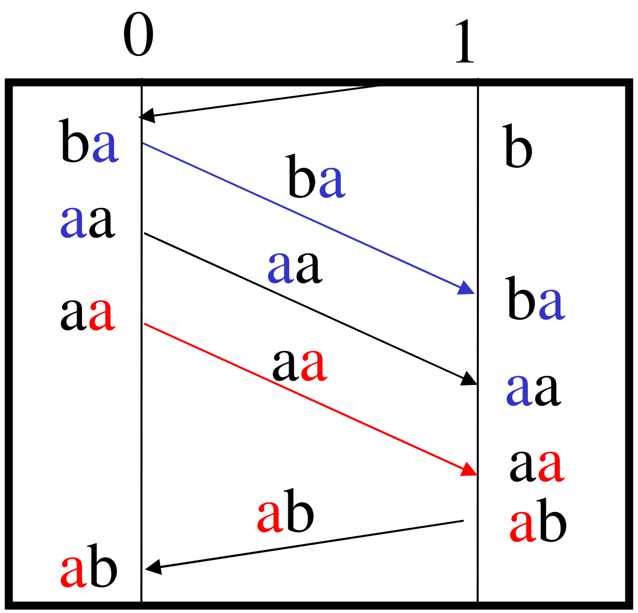
b =

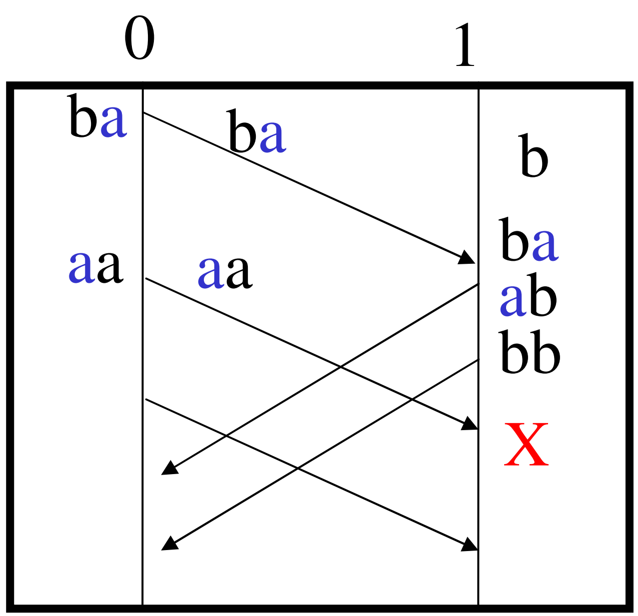


(a,b)* can be implemented with deterministic additional information.

Pb: Cannot remember all the scenario (infinite states)

remember last actions+current action!





Example of a specification hard to distribute

Distributed Implementation:
Each process = Finite Automaton A_i

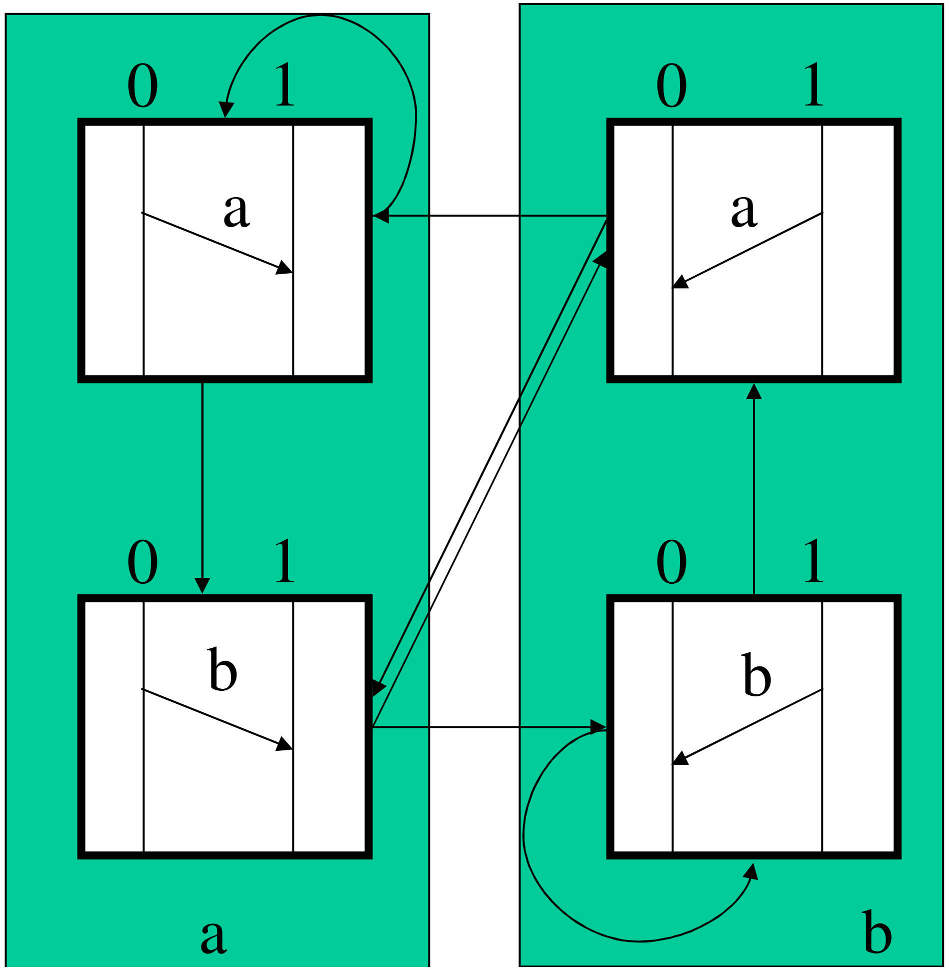
different setting:
State of each process:

- attached to mesages sent by the process
- determined non deterministically

=> Can make choices for others.

Example of a specification hard to distribute

Process can add a non deterministic bounded information with their messages. => can make choices.
Here, process announces next scenario.



More powerful but sometimes not good implemetation (ex: client chooses whether server grants him access)

Different settings:

Easier

a) Nothing added to messages
b) Deterministic additional information
c) Non deterministic additional information

Less hypothesis

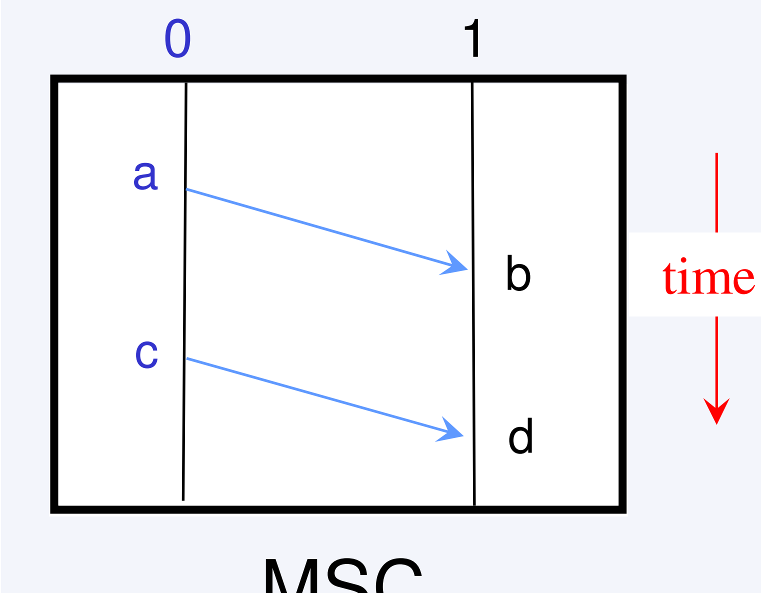
Specification and Implementation Models

[Genest, Muscholl, Peled :

Survey 2003/2004 in Concurrency and Petri Nets 2003]

Message Sequence Charts (MSCs)

Widely used: TelCo companies, UML sequence diagram, ITU norm, SDL
Also in distributed algorithms etc.



MSC

Partial order $\leq$ on events a,b,c…

b,c incomparable

Process Order: $a <_0 c$

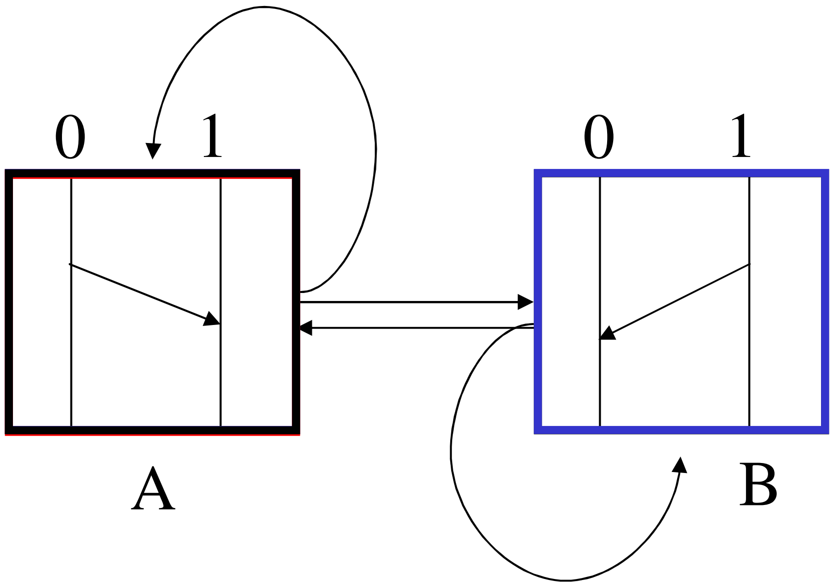
Message Order: $a < b$
(1st send from 0 to 1 received
By 1st receive on 1 from 0)

Total order = linearization = execution
a b c d or a c b d

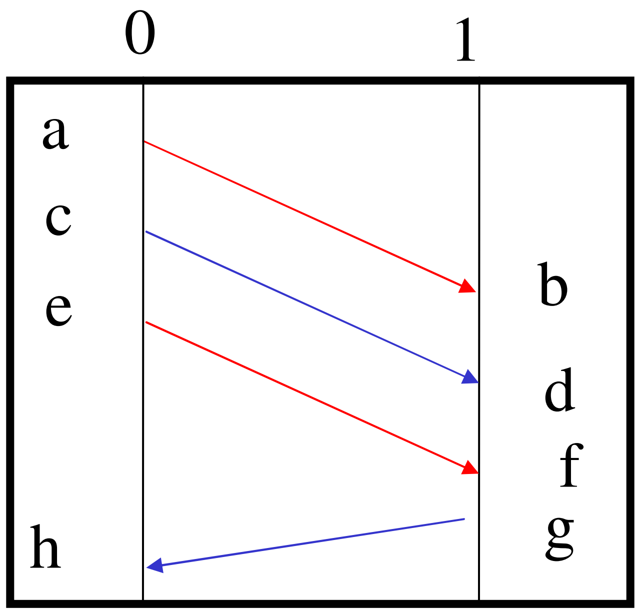
Any linearization w=> unique MSC M_w : define $[w] = \{v \mid M_v = M_w\}$

MSCs-graphs

Graphes whose nodes are MSCs = Rational languages of Scenarios



Composition: glue scenario
Along same process line

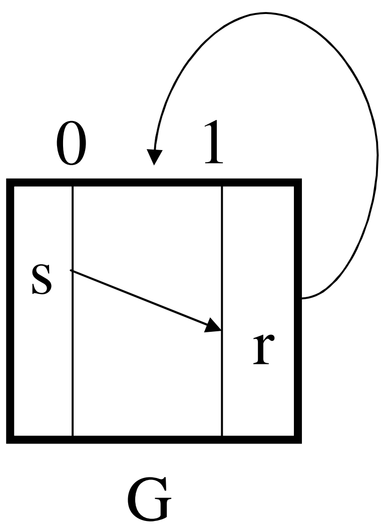


b (first scenario) can happen (in time) after e (third scenario)

a b c d e f g h   but also   a c e b d f g h

MSCs-graphs

Graphes whose nodes are MSCs = Rational languages of Scenarios

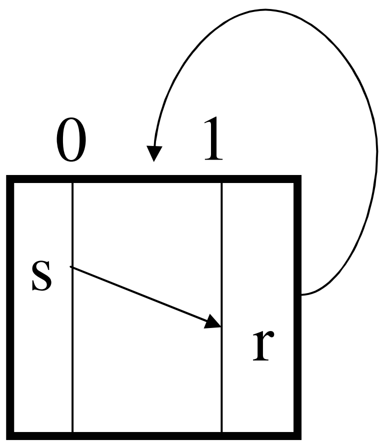


Define language L(G) as set of executions of MSCs.

QUIZZ: Regular or not? What is L(G)?

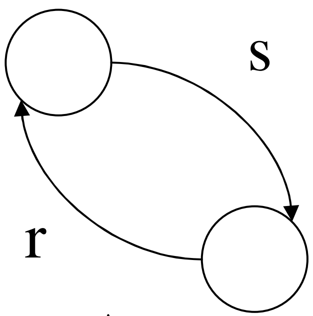
MSCs-graphs

Graphes whose nodes are MSCs = Rational languages of Scenarios



G

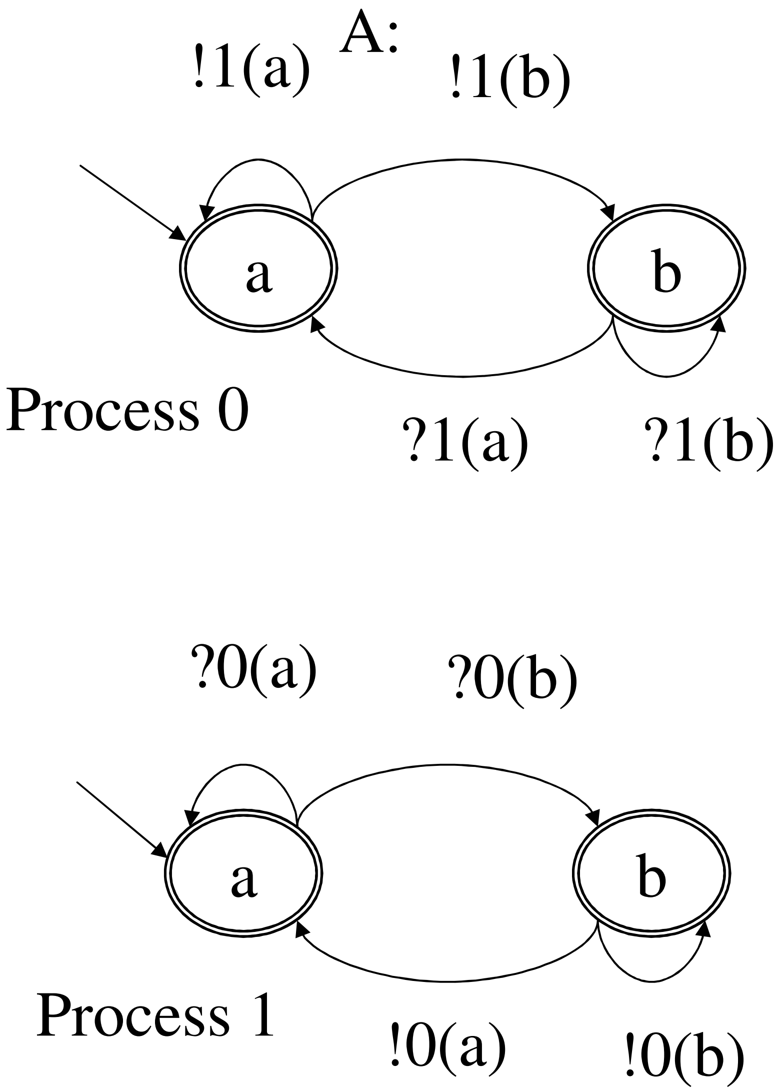
Define language L(G) as set of executions of MSCs.



A_G

Define automaton A_G by choosing a linearisation of the scenario in the node.
$L(A_G)$ is called a set of representatives for L(G).
Then $L(G) = [L(A_G)]$, closure of a regular language.

Communicating Automaton



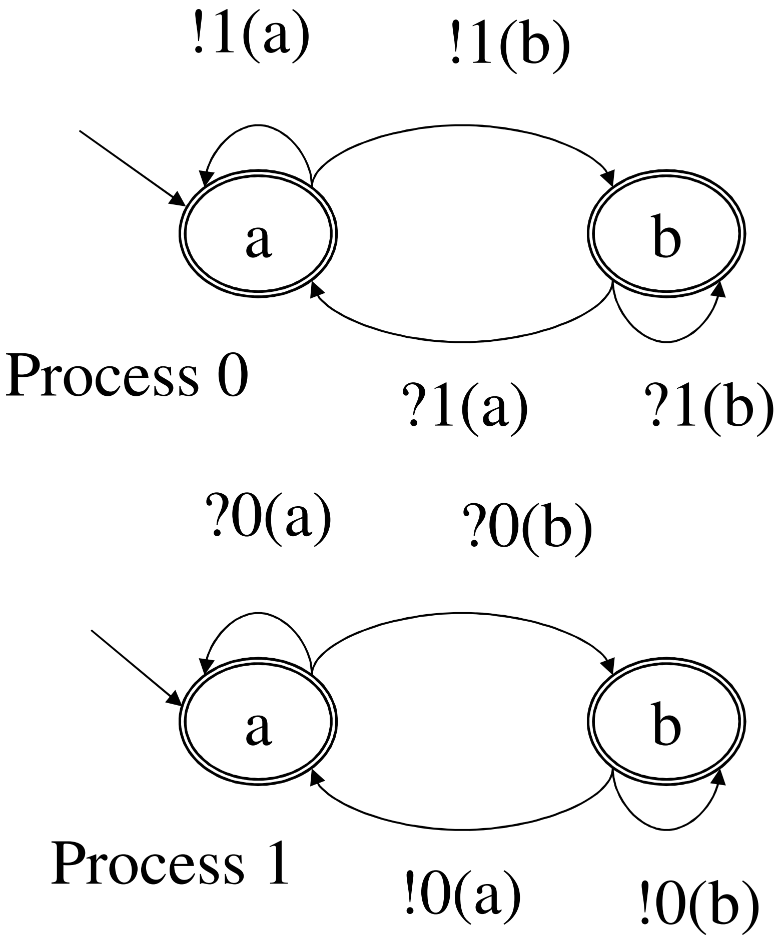
One finite state automata
For each process.

Actions: sends from another process
and receives from another process
with content.

Implicit: communication buffers

Communicating Automaton

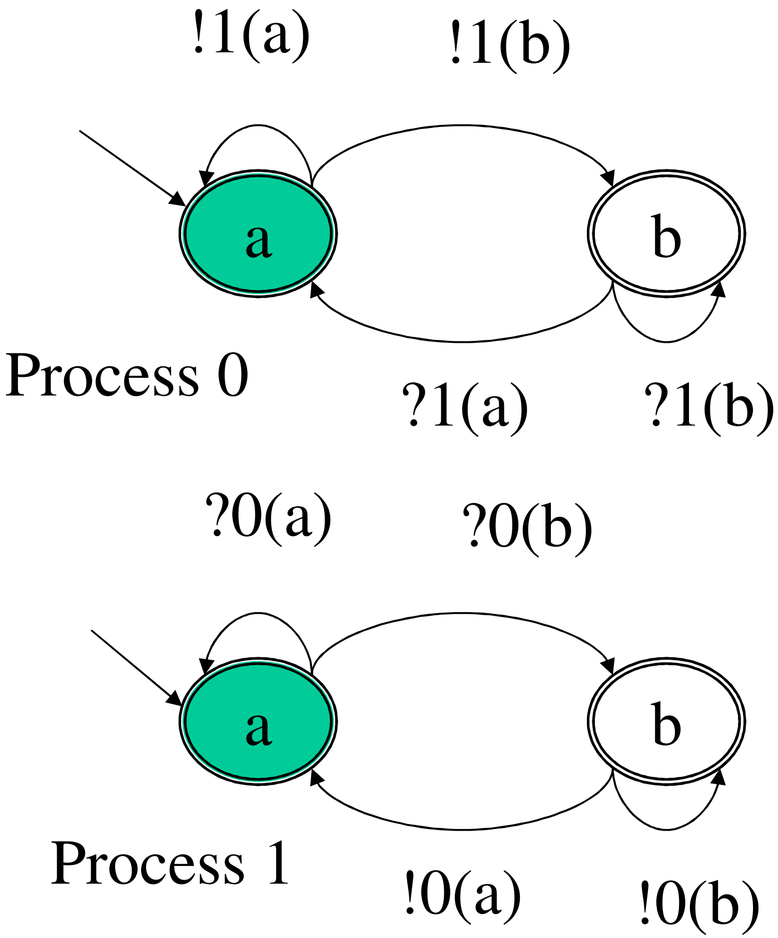
A:

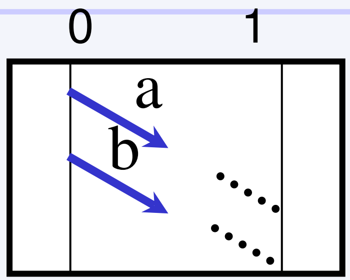


Configuration: states of 0,1 and
Buffer content (tuple of words)

Communicating Automaton

A:





Implicit FIFO buffers

Configuration: states of 0,1 and Buffer content (tuple of words)

Execution (forget additional data):

0!1 0!1 1?0 1?0

L(A): set of executions possible (reaching final states+empty buffer)

0!1 1?0 0!1 1?0

Realizability

Realizability question: Given MSC-graph G,
find Communicating automaton A with L(A)=L(G) (if possible).

Notice: $[L(A)] = L(A)$ and $[L(G)] = [[L(A_G)]]=[L(A_G)] = L(G)$
Both are closed by commutation, none are regular in general.

Realizability

Realizability question: Given MSC-graph G,
find Communicating automaton A with L(A)=L(G) (if possible).

Notice: There are non regular MSC-graphs that can be realized

QUIZZ: give an example.

Realizability

Realizability question: Given MSC-graph G,
find Communicating automaton A with L(A)=L(G) (if possible).

Notice: There are MSC-graphs that cannot be realized.

QUIZZ: give an example.

Restrictions, Regularity and more

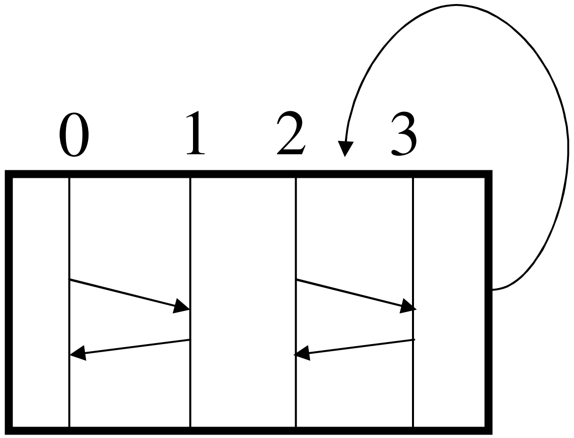
[Rajeev Alur, Mihalis Yannakakis: CONCUR 99]

[Anca Muscholl, Doron Peled : MFCS 99]

[Blaise Genest, Anca Muscholl, Helmut Seidl, Marc Zeitoun: ICALP 2002 & JCSS 2006]

[Genest, Kuske, Muscholl : Fundamentae Informaticae 2007]

Restrictions



G

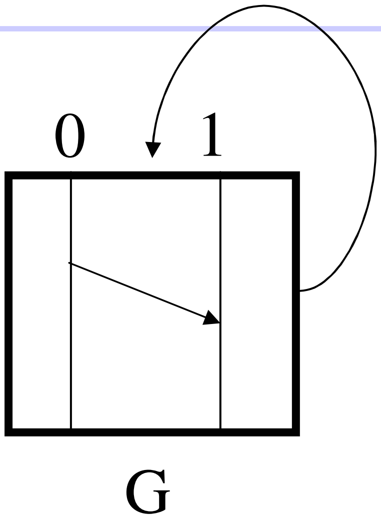
Cannot be realized

Communicating graph of a MSC: 1 node per process, 1arrow if message

Not weakly connected

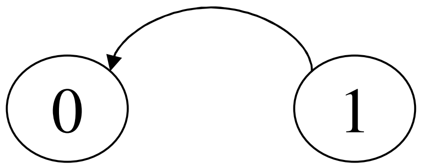
globally-cooperative MSC-graphs:
Each loop (strongly connected component) of G, is weakly connected
regular MSC-graphs (also called bounded):
Each loop (strongly connected component) of G is strongly connected

Restrictions



Can be realized

Communicating graph of a MSC: 1 node per process, 1arrow if message



weakly connected

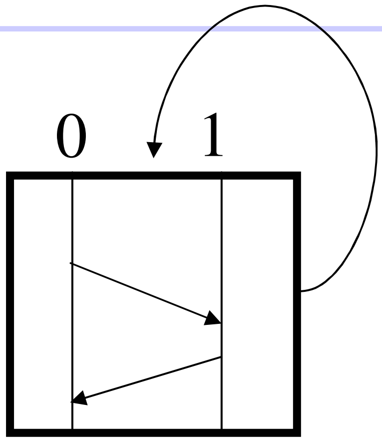
globally-cooperative MSC-graphs:

Each loop (strongly connected component) of G, is weakly connected

regular MSC-graphs (also called bounded):

Each loop (strongly connected component) of G is strongly connected

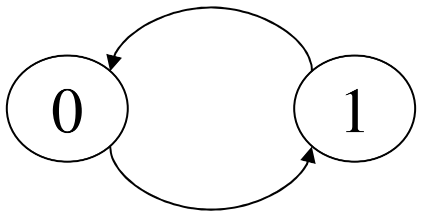
Restrictions



G

Can be realized

Communicating graph of a MSC: 1 node per process, 1arrow if message



strongly connected

globally-cooperative MSC-graphs:

Each loop (strongly connected component) of G, is weakly connected

regular MSC-graphs (also called bounded):

Each loop (strongly connected component) of G is strongly connected

Restrictions

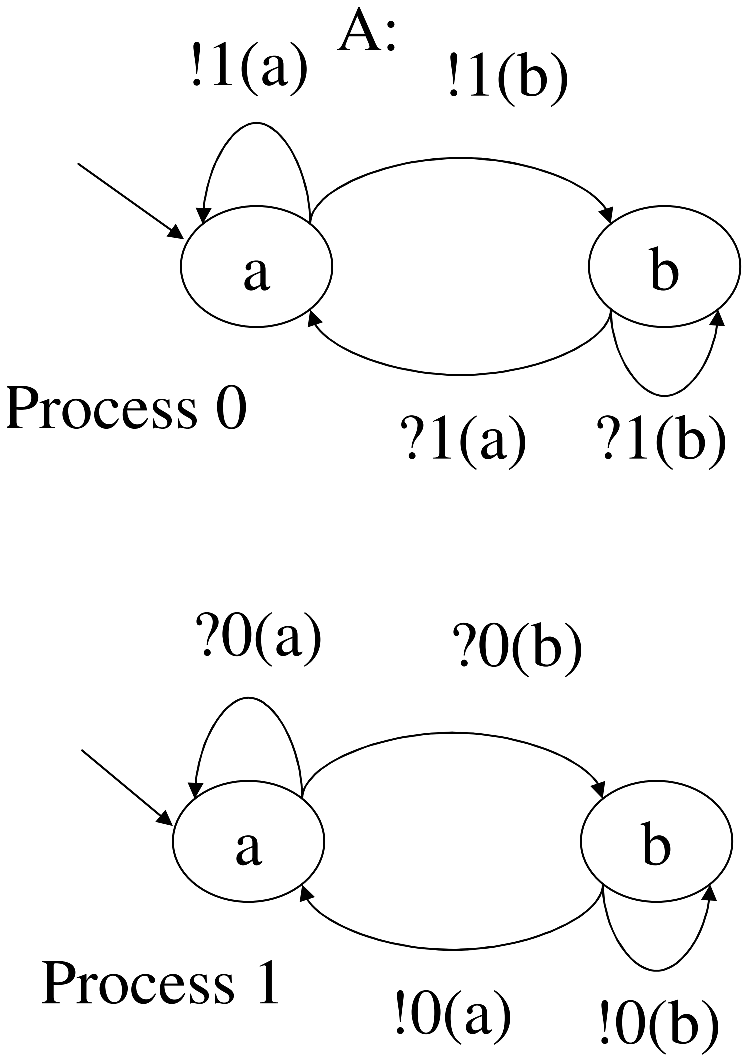
globally-cooperative MSC-graphs:
Each loop (strongly connected component) of G, is weakly connected
regular MSC-graphs (also called bounded):
Each loop (strongly connected component) of G is strongly connected

If G is a regular MSC-graph, then L(G) is regular.

If G is a globally cooperative MSC-graph, then for all B,
Set of B bounded executions of L(G) is regular.

Testing any restriction is co NP-complete

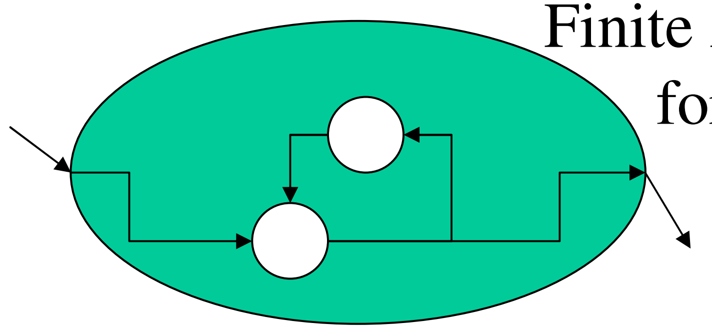
Regularity



communicating automaton A

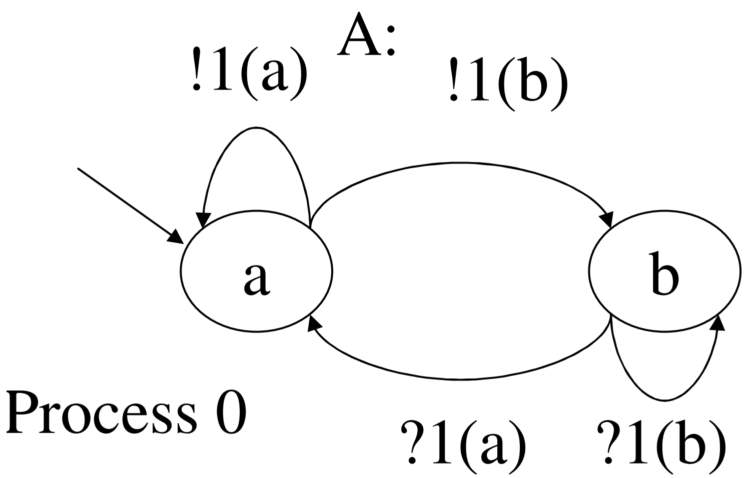
L(A) is regular iff channels
are bounded: there exists B,
for all w $\in$ L(A),for all prefix v of w:
Number of sends in v
$\leq$ B + number of receives in v

Finite Automaton
for L(A)



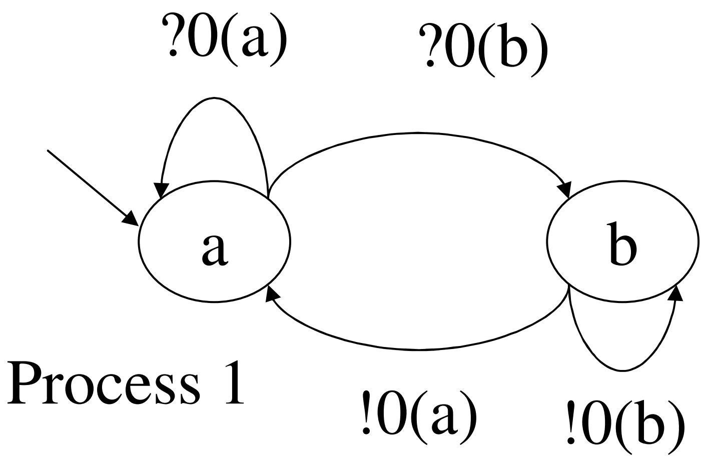
Loop: Same number of sends and receives

Regularity



communicating automaton A

L(A) is regular iff channels are bounded: there exists B, for all w $\in$ L(A),for all prefix v of w: Number of sends in v $\leq$ B + number of receives in v

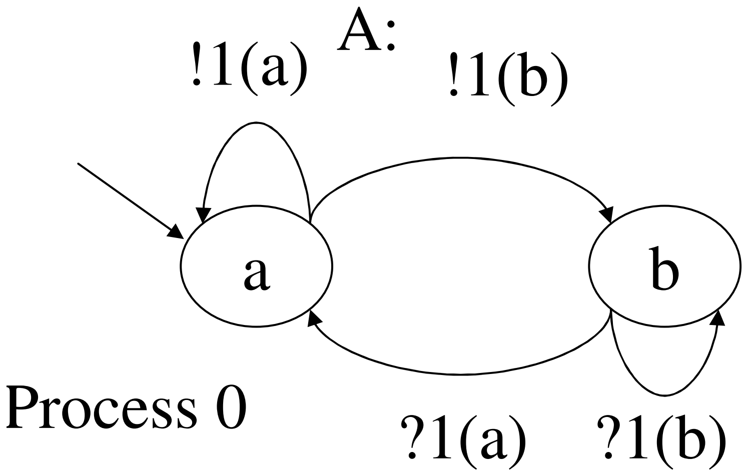


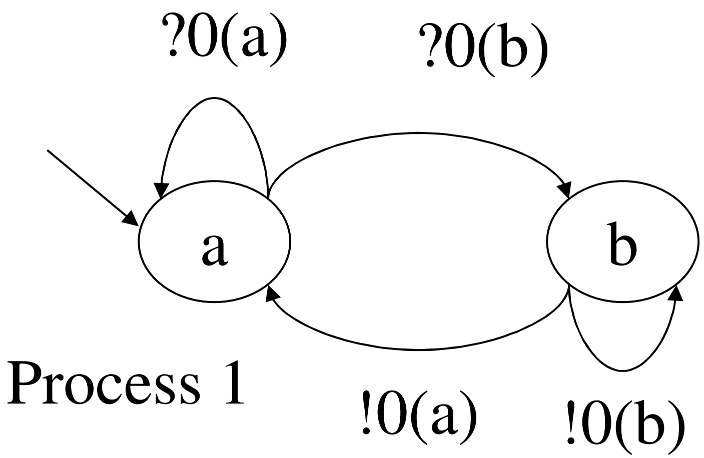
Undecidable to test if L(A) is regular (=bounded)

Undecidable to test if L(A) is B-bounded

(comm. Automata are Turing powerful: reduction to L(A)= $\emptyset$)

Regularity





Decidable to test if $L^{pref}(A)$ is B bounded (L^{pref} considers all states final and accepts even if some message not yet received)

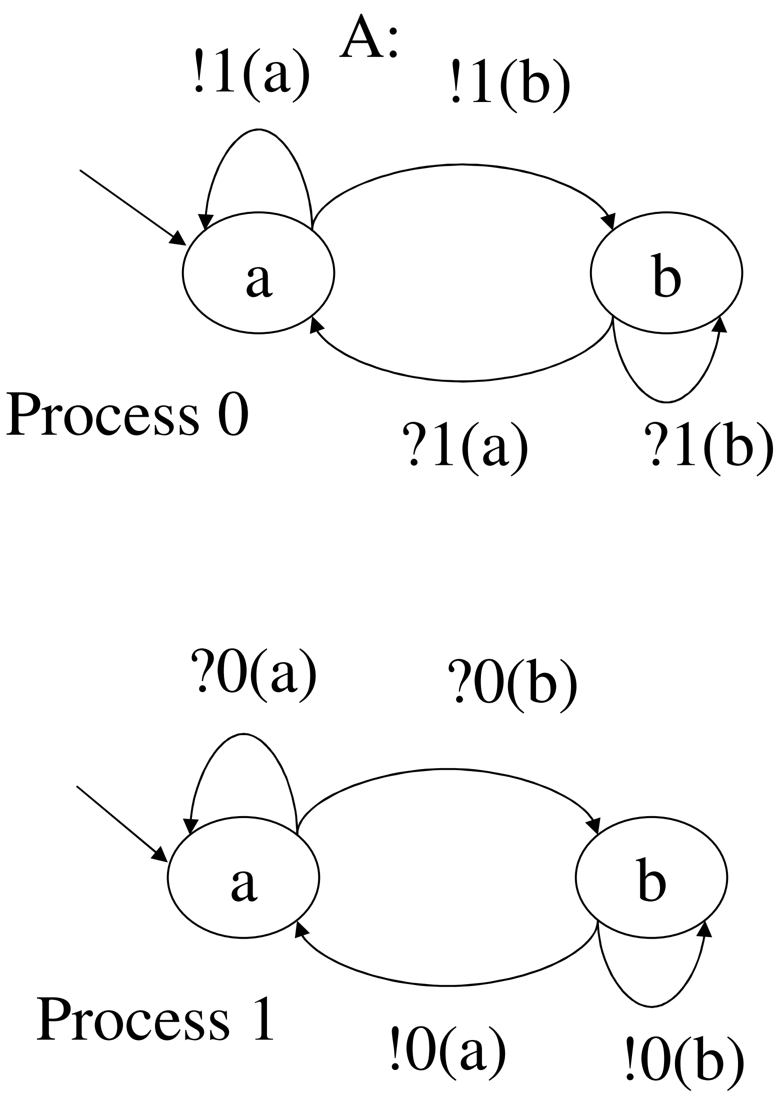
-Construct L^{pref} (A) up to bound B+1.

-Check whether bound B+1 is reached by an execution.

This execution is in $L^{pref}(A)$ and is not B bounded.

Else $L(A) \subseteq L^{pref}(A)$ are B-bounded

Deterministic Communicating Automata



deterministic if when
2 transitions from same state
labeld by !p(m) and !p(n),
then m=n.

QUIZZ: deterministic?

Realizability

Realizability question: Given MSC-graph G,
find Communicating automaton A with L(A)=L(G) (if possible).

a) No message content
b) Deterministic additional information
c) Non deterministic additional information

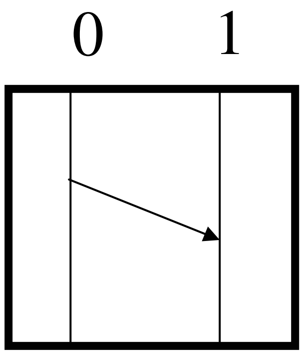
No message content

[Rajeev Alur, Kousha Etessami, Mihalis Yannakakis: ICALP 2001 & TCS 2003]

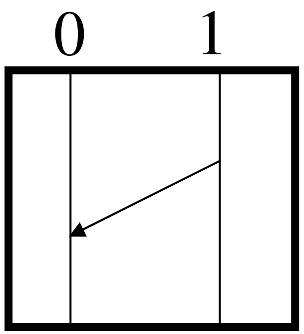
[Markus Lohrey : CONCUR 2002 & TCS 2003]

Example of a specification hard to distribute

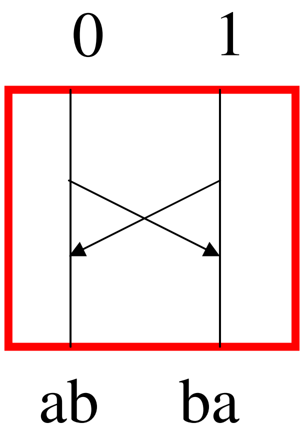
a =



b =



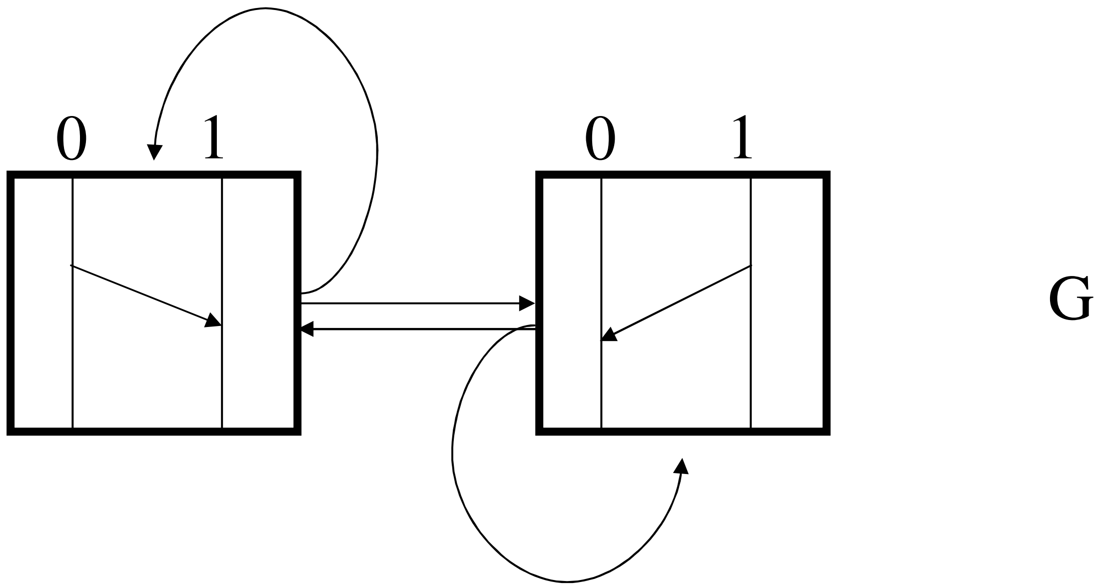
c =



(ab or ba) not possible to implement with
no information exchange between processes

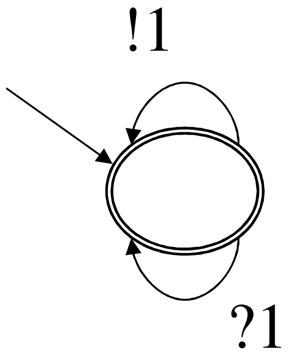
c looks like something legitimate for each process.

Intuition

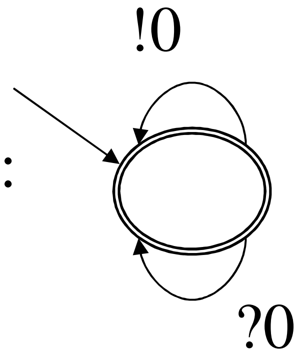


Look at the projection A_0,A_1 of G on each process 0 and 1.
Both A_0,A_1 are regular language => communicating automaton A.

A_0:



A_1:



Claim:
G is realizable iff
L(A)=L(G), and then
A is an implementation.

Proof

Claim: G is realizable iff L(A)=L(G), and then A is an implementation.

Assume that $\exists$ communicating automata with local final states B with L(B)=L(G)

By construction, L(G) $\subseteq$ L(A). Let us show that L(A) $\subseteq$ L(G)
Let w $\in$ L(A).

Then for all p, $\pi_p(w) \in L(A_p)= L(\pi_p(G))$:
$\exists$ x $\in$ L(G)=L(B) with $\pi_p(x) = \pi_p(w)$.
It means that $\pi_p(x)=\pi_p(w)$ reaches a final state of B_p.

Hence w reaches a final state on every process of B_p:
w $\in$ L(B)=L(G).

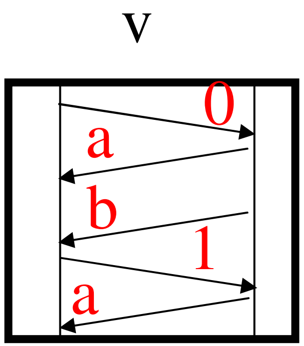
Undecidability

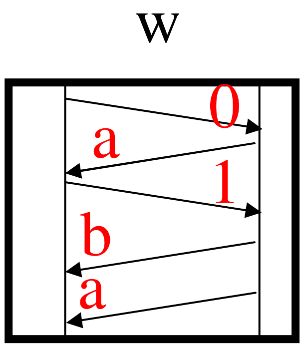
Claim: G is realizable iff L(A)=L(G), and then A is an implementation.

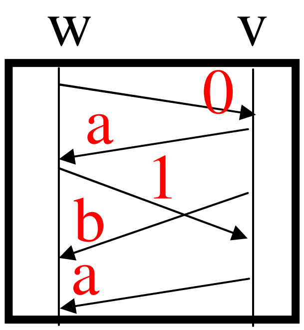
Theorem: Checking whether L(G) is realizable is undecidable
(even if G is regular and $L(A_G)=L(G)$)

Reduction to PCP: words (v_i,w_i) on $\{a,b\}^*$

Ex: $(v_0,w_0)=(ab,a)$ and $(v_1,w_1) = (a,ba)$







$\in L(A)$
$\notin L(G)$

PCP has a solution iff not realizable

Some Decidability results

Claim: G is realizable iff L(A)=L(G), and then A is an implementation.

Theorem: Checking whether L(G) is realizable is co-NEXPTIME when G has no loop (finite set of MSCs).

Theorem: Checking whether Pref(L(G)) = $L^{pref}(A)$ is EXPSPACE-complete when G is globally-cooperative (includes regular MSC-graphs). In general, undecidable.

Intuition

Theorem: Checking whether $Pref(L(G)) = L^{pref}(A)$ is EXPSPACE-complete when G is globally-cooperative (includes regular MSC-graphs). In general, undecidable.

Intuition: $L^{pref}(A) = Pref(L(G))$?

Regularity of G gives a bound on number of messages in transit for executions of G.

$\Rightarrow$Check whether all executions of $L^{pref}(A)$) are bounded.

If not, $L^{pref}(A) \neq Prefix(L(G))$.

If yes, test equality of two regular sets ($L(G)$ exponential in $|G|$).

deterministic additional information

[Henriksen, Madhavan Mukund, Narayan Kumar, Mihind Sohoni, P.S.Thiagarajan : CONCUR-ICALP 2000; I&C 2005]

[Dietrich Kuske STACS 2002, I&C 2004]

Henriksen et Al. Theorem

Deterministic bounded Communicating automaton

Difficult part

All these regular (thus bounded) formalisms are equivalent. Every regular MSC graphs are realizable!

regular CMSC-graph

regular set of linearizations

MSO formula that are bounded

Mazurkiewicz trace Theory

Asyncrhonous
Automaton

Zielonka
Theorem '87

Don't reinvent the wheel,
lift known results.
Reasoning with messages is hard,
try to break down problem

Ochmanski theorem '83

Loop connected
Rational
set of traces

Regular set of
traces

MSO
formula
on traces

Mazurkiewicz trace Theory

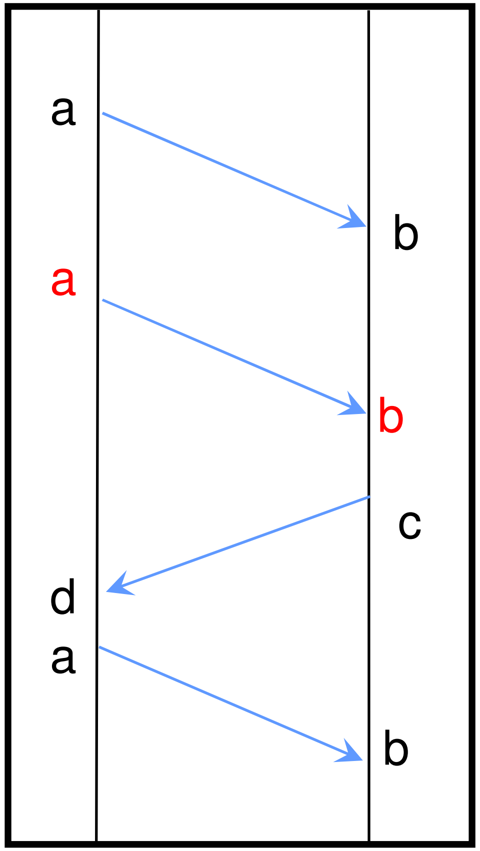
Given: Alphabet Σ and Symetric Independance relation $I \subseteq \Sigma \times \Sigma$

Define $\sim$ smallest equivalence relation containing
uabv $\sim$ ubav when $(a,b) \in I$

Traces are equivalence class of $\sim$ over words of Σ^*

Ideas: words $\Leftrightarrow$ linearizations/executions
Traces $\Leftrightarrow$ MSCs
Encode commutations of MSCs into fixed Independance alphabet Σ,I

Kuske’s Alphabet



MSC 2-bounded

Alphabet: Ω = a,a,b,b,c,c,d,d

commutation

a b $\sim_{\mathcal{I}}$ b a
a b $\sim_{\mathcal{I}}$ b a
a c $\sim_{\mathcal{I}}$ c a
d b $\sim_{\mathcal{I}}$ b d

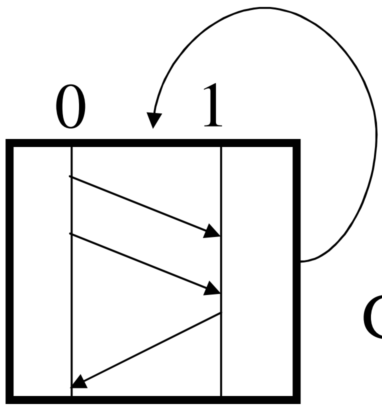
ab ab cd $\sim_{\mathcal{I}}$ a a b b c d

Lin(M)=[ab ab ab]$_{\mathcal{I}}$

red a can be received only by a red b

Kuske's Alphabet

L(G) is regular and
2 bounded



Representatives on Σ

$G \longrightarrow A_G \longrightarrow C_G$

On Ω

Alphabet: Ω= a,a,b,b,c,c,d,d
(a=0!1, b= 1?0, c=1!0,d=0?1)

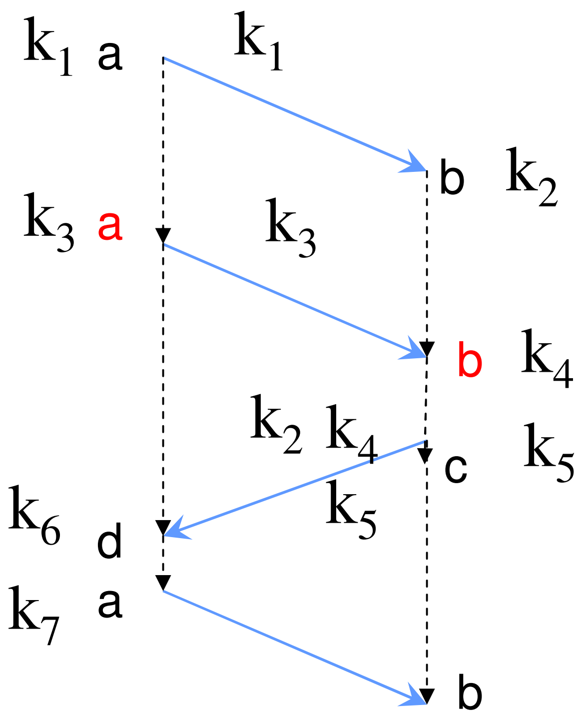
a b $\sim_{\mathcal{I}}$ b a a c $\sim_{\mathcal{I}}$ c a c d $\sim_{\mathcal{I}}$ d c
a b $\sim_{\mathcal{I}}$ b a d b $\sim_{\mathcal{I}}$ b d d c $\sim_{\mathcal{I}}$ c d

$$L(G) = \pi_{\Sigma}[L(C_G)]_{\mathcal{I}}$$

C_G is loop connected since loops of G are strongly connected.
Ochmanski: L(G) is regular. Zielonka: $\exists$ AA, L(AA)=L([C_G])

Asynchronous Automaton simulation

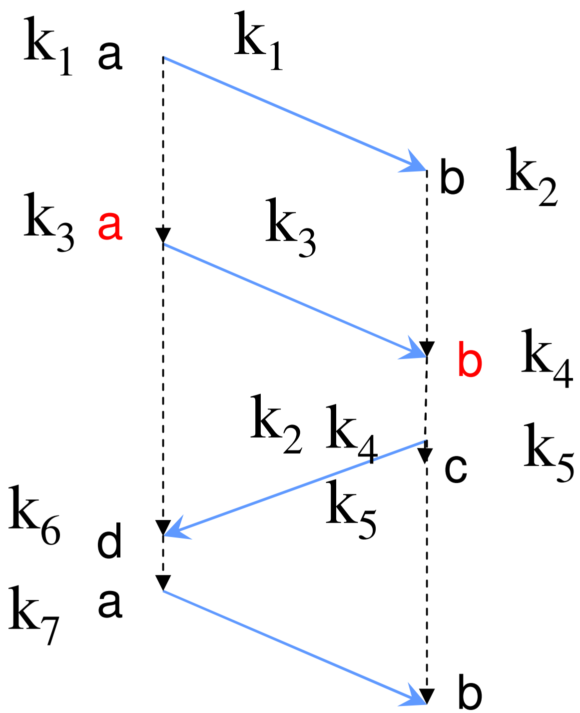
Simulation by a communicating automaton A
of deterministic Asyncrhonous Cellular Automaton AA:
each event is labeled by a state $k \in K$ (K finite set)
new state depends only upon states of dependent letters



For each letter, local state remembers state of last event with that letter.

k_7 computed using k_2

Communicating automaton A



Final states of communicating automaton A
= final states of asynchronous automaton AA

For each B bounded linearization w,
w $\in$ L(A) iff w $\in$ L(AA).

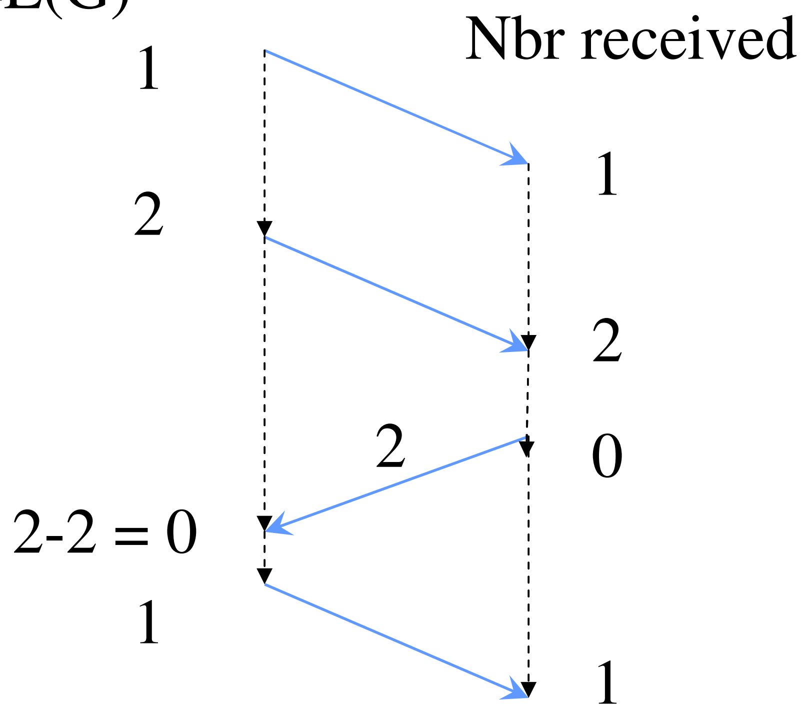
Set of B bounded lin.

So L(A) $\cap \Sigma^*_B$ = L(AA) $\cap$ Σ_B*= L(G) $\cap \Sigma_B$*=L(G)

Communicating automaton A

So $L(A) \cap \Sigma^*_B = L(AA) \cap \Sigma_B^* = L(G) \cap \Sigma_B^* = L(G)$

Build A_B with $w \in L(A_B)$ iff M_w is B bounded.
Then $L(A x A_B) = L(G)$

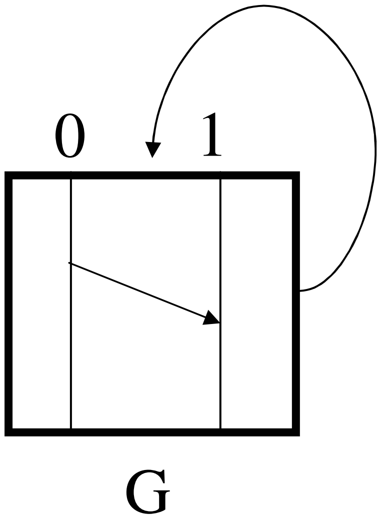


Idea: Each process counts how many messages in transit up to B+1.
If reaches B+1 at some point, then reject, else accepts.

Non deterministic additional information

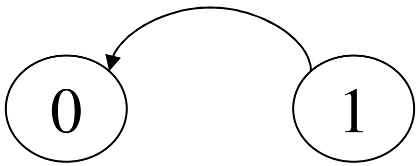
[Blaise Genest, Dietrich Kuske, Anca Muscholl I&C 2006]

Globally cooperative MSC-graphs



Can be realized

Communicating graph of a MSC: 1 node per process, 1arrow if message



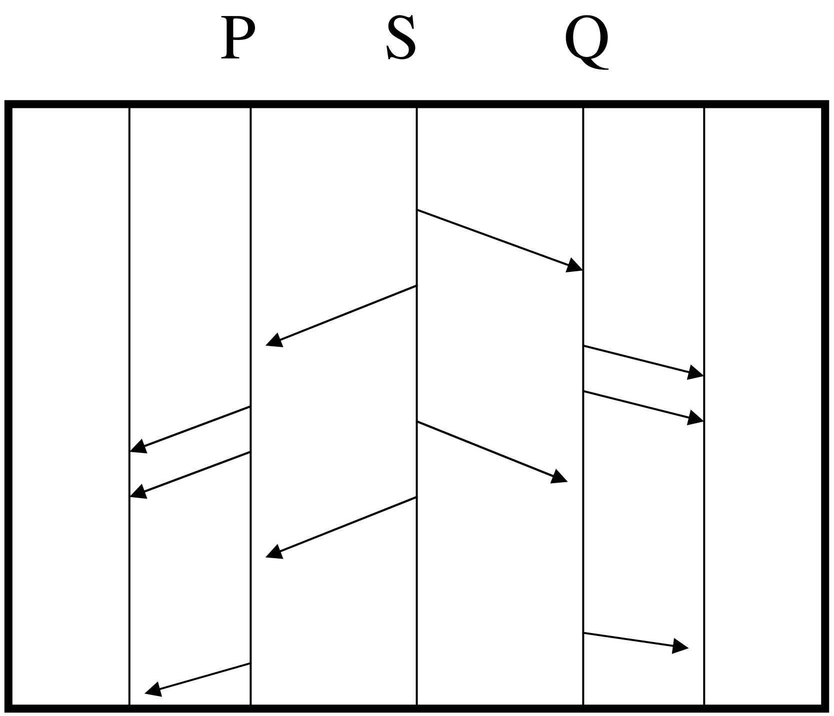
weakly connected

Globally cooperative but not regular and still realizable.

Idea: Extends Henrinksen et al.

Globally cooperative MSC-graphs

There are globally cooperative MSC-graphs which cannot be realized with deterministic communicating automata.



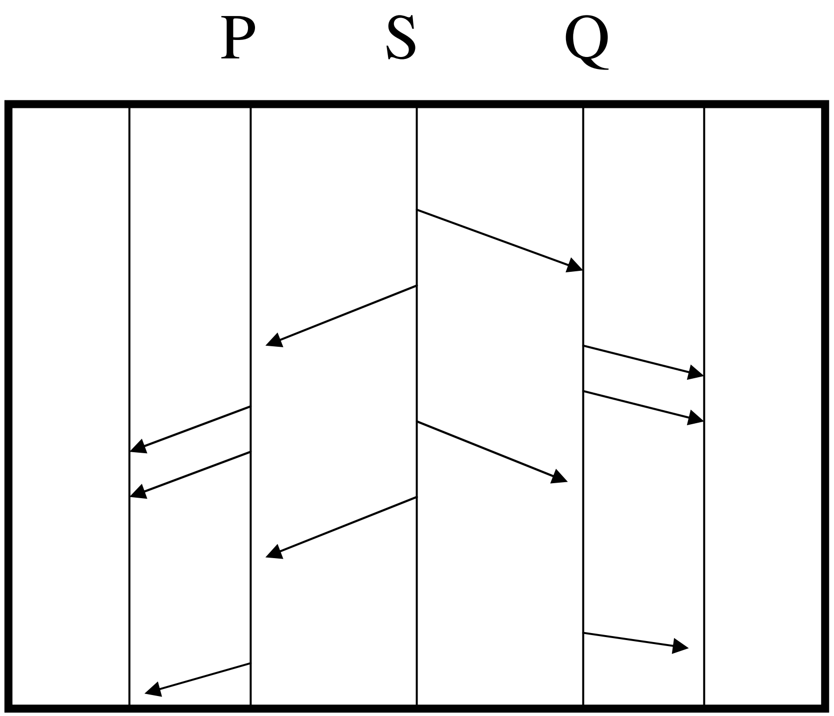
Spec:
After the n-th receive of S,
process P sends $p(n)$ messages.
process Q sends $q(n)$ messages.

$p(n)=q(n) \in \{1,2\}$

Easy to implement with non determinism (S chooses $p(n)=q(n)$).

Globally cooperative MSC-graphs

There are globally cooperative MSC-graphs which cannot be realized with deterministic communicating automata.



Assume A deterministic implementing G.
A has n states on each process.

There exists two sequences $(a_i);(b_i)$, $i \in \{1.. \ln(n^5)\}$ of p(n) after which states of all process are the same.
A has to accept mix of (a_i) on P and (b_i) on Q, contradiction

Globally cooperative MSC-graphs

Need candidates for class of Communicating automata equivalent with globally cooperative MSC-graphs.

A is said « existentially bounded» if ∃ regular set of representative.

All MSC-graphs are existentially bounded (Remember A_G).
Globally cooperative have furthermore $Lin^B(A)$ regular set of representatives

An ∃ bounded communicating automaton A has $Lin^B(A)$ regular set of representatives.

Genest et Al. Theorem

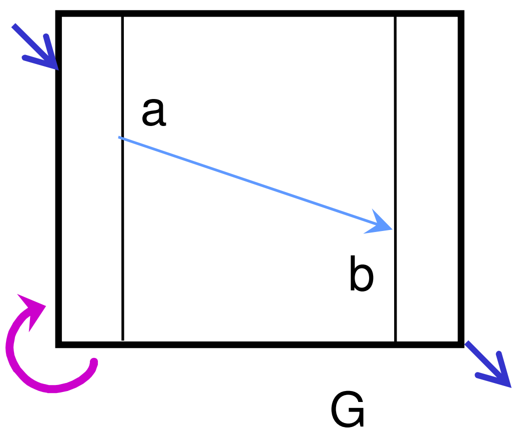
(Non –Deterministic)
$\exists$ bounded
Communicating
automaton

Globally
cooperative
CMSC-graph

$Lin^B(G)$ regular
set of
representatives

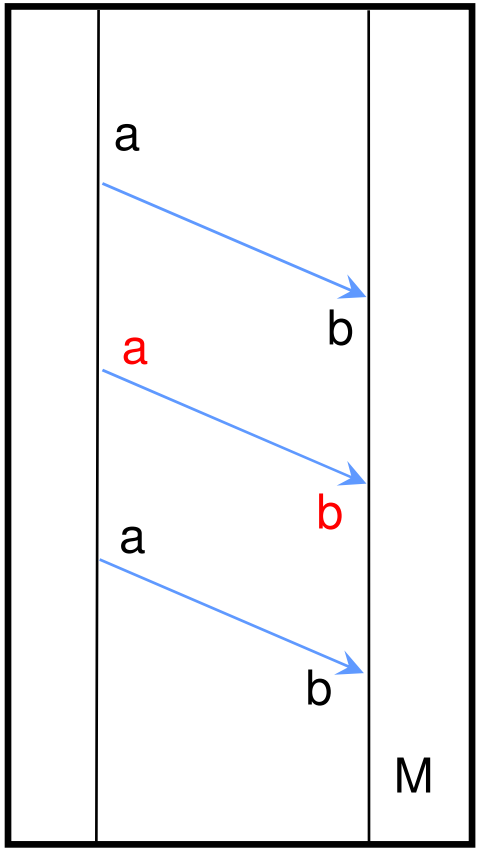
MSO
formula
that are
$\exists$ bounded

Reuse Same Kuske Encoding!



G is ∃-2 bounded.

Reuse same Kuske Encoding!



Alphabet: $\Omega = a, a, b, b$

commutation $a\, b \sim_{\mathcal{I}} b\, a$
$a\, b \sim_{\mathcal{I}} b\, a$

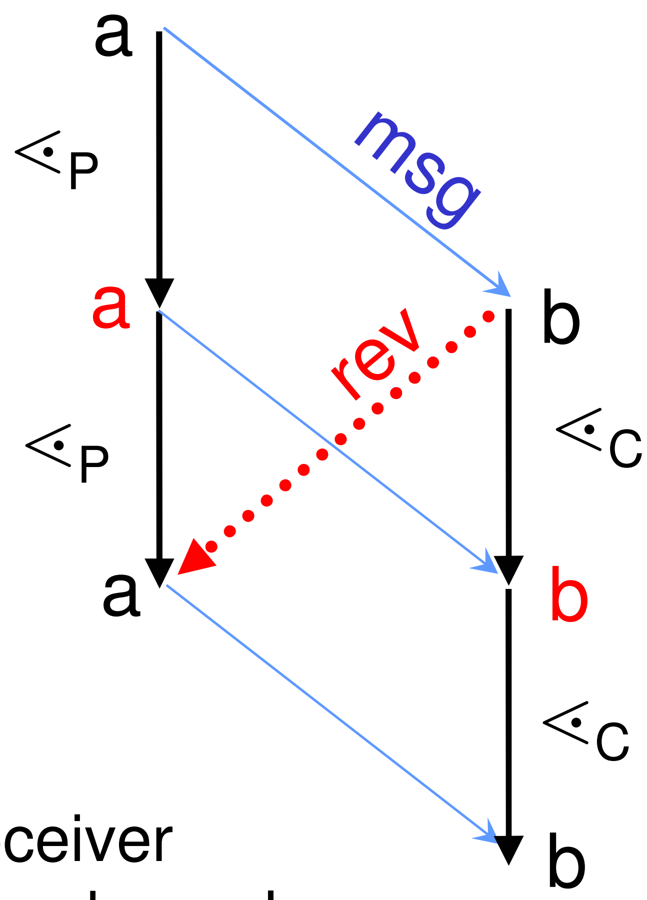
$ab\ ab\ ab \sim_{\mathcal{I}} a\ a\ b\ a\ b\ b \not\sim_{\mathcal{I}} a\ a\ a\ b\ b\ b$

$Lin^2(M) = [ab\ ab\ ab]_{\mathcal{I}}$

Trace alphabet gives us only B bounded linearizations, not all

Kuske Encoding

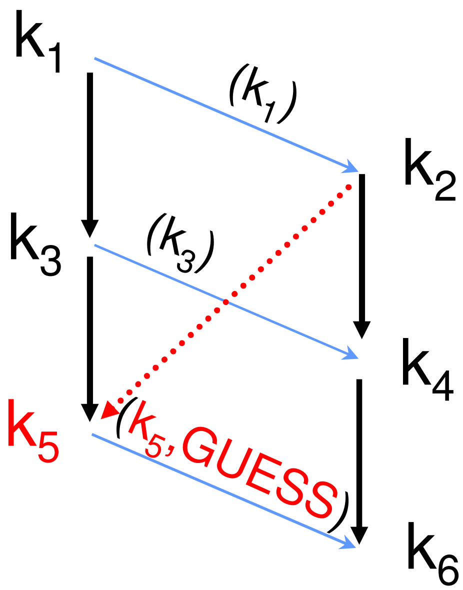
Associated
Partial order:



rev associate i-th receiver
With i+Bth send on same channel

Simulation of the Cellular AA

Simulate with non deterministic Communicating automaton:
Will guess value for rev, and check them later.



test that GUESS=k_2

To compute k_5, needs value of k_2: no way to know it for sure:
Guess it, and chek it later.

Communicating automaton A

$$L(A) \cap \Sigma^*_B = L(AA) \cap \Sigma_B^* = L(G) \cap \Sigma_B^* = L(G)$$

Build A_B with $w \in L(A_B)$ iff M_w has a B bounded linearization.
Then $L(AxA_B)=L(G)$

Idea: New relation $\boxed{\begin{array}{c} e \lessdot_a f \text{ if } e <_p f \text{ and} \\ f \text{ first event of type } a \end{array}}$

M is $\exists$–B-bounded iff $(rev \cup msg \cup \lessdot_a)$ is acyclic

Prop: If cycle in $(rev \cup msg \cup \lessdot_a)$, then a cycle of bounded size

look for cycle in this relation (need to guess for rev).

Conclusion and Future Work

Realizability: Many settings, many results.
This talk: only a glimpse of all the results.

Still too expansive (time to check, size of the implementation)
or too restricted (class to implement from).

Other results:
Weaker results for very generic system (local EMSO) and
Small non deterministic implementation for very restricted systems
(local choice MSC-graphs, incomparable with regular MSC-graphs)

In term of techniques: lift result of simpler specifications (Traces).

Future work: handle distributed games (non controllable events,
specification is only set from which to choose strategy),
Main problem generate a distributed strategy.