Guessing the buffer bound for k-synchronizability

GT ALGA - Videoconference

Cinzia Di Giusto, **Laetitia Laversa** and Etienne Lozes

June 17, 2021

Université Côte d'Azur - Laboratoire I3S - Sophia Antipolis, France

Distributed systems

- **Distributed System**: several machines that cooperate to achieve a common goal
- Systems are modeled as **Communicating Automata**
- **Synchronous** Communication: sending and receiving happen simultaneously
- **Asynchronous** Communication: receiving is separated from sending
    - FIFO Buffers in mailbox configuration

Synchronizability - Motivation

Asynchronous CA are Turing equivalent

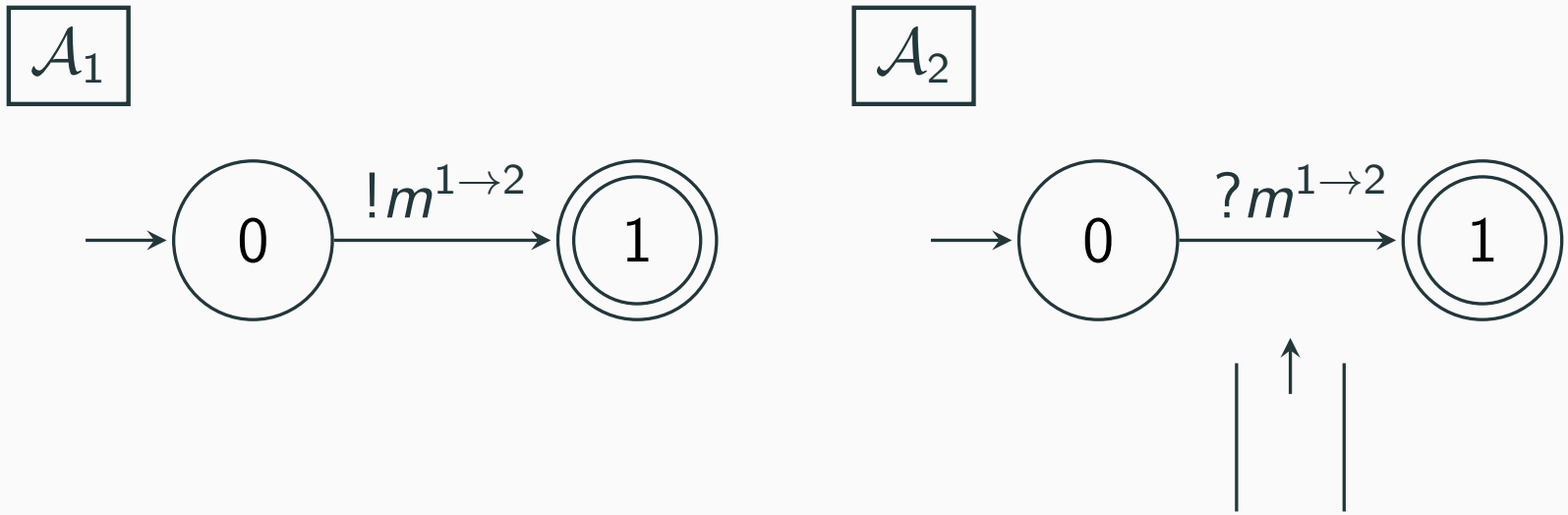
$\Rightarrow$ How can we mimic synchronous communication while keeping some degrees of freedom?

- Existentially k-bounded [Lorhey *et al.*, 2004]
- Synchronizability [Basu *et al.*, 2016]
- k-synchronizability [Bouajjani *et al.*, 2018]

How to model distributed systems

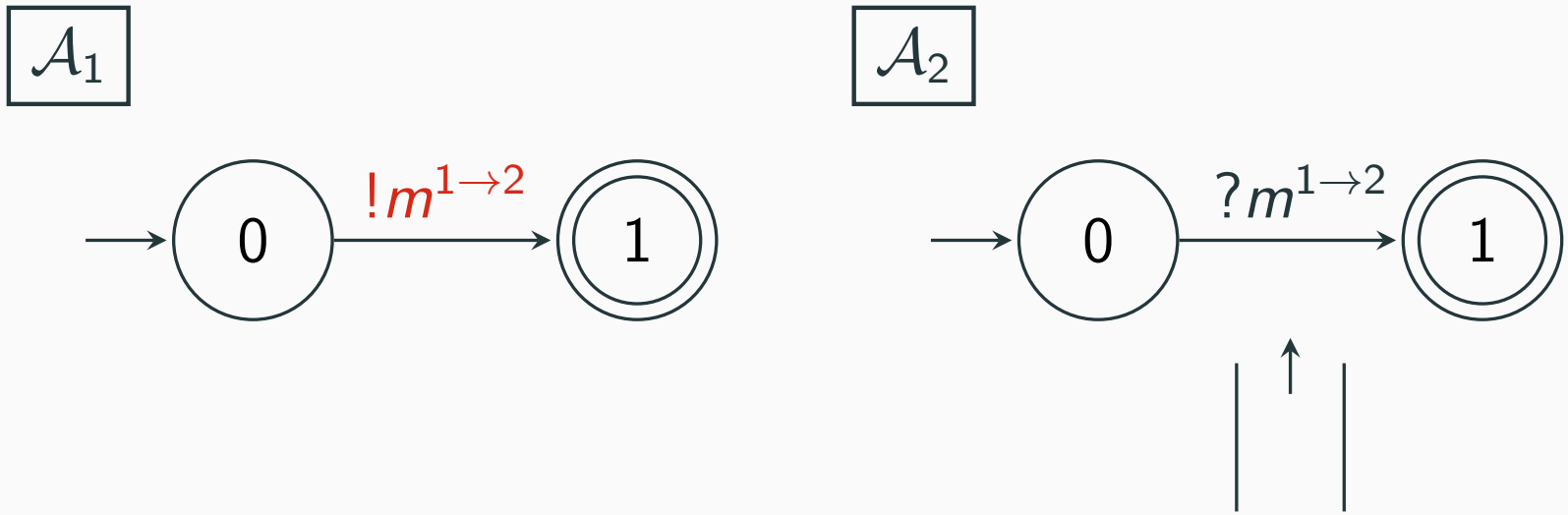
Communicating Automata

Finite state automata enriched with buffers to store exchanged messages



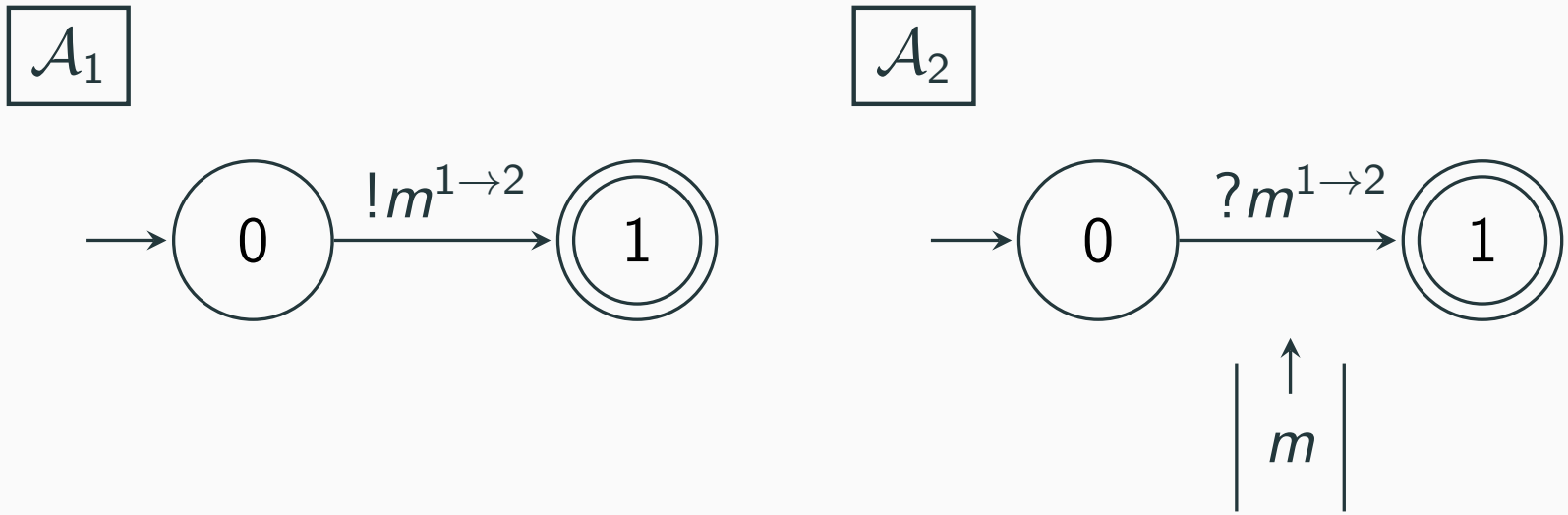
Communicating Automata

Finite state automata enriched with buffers to store exchanged messages



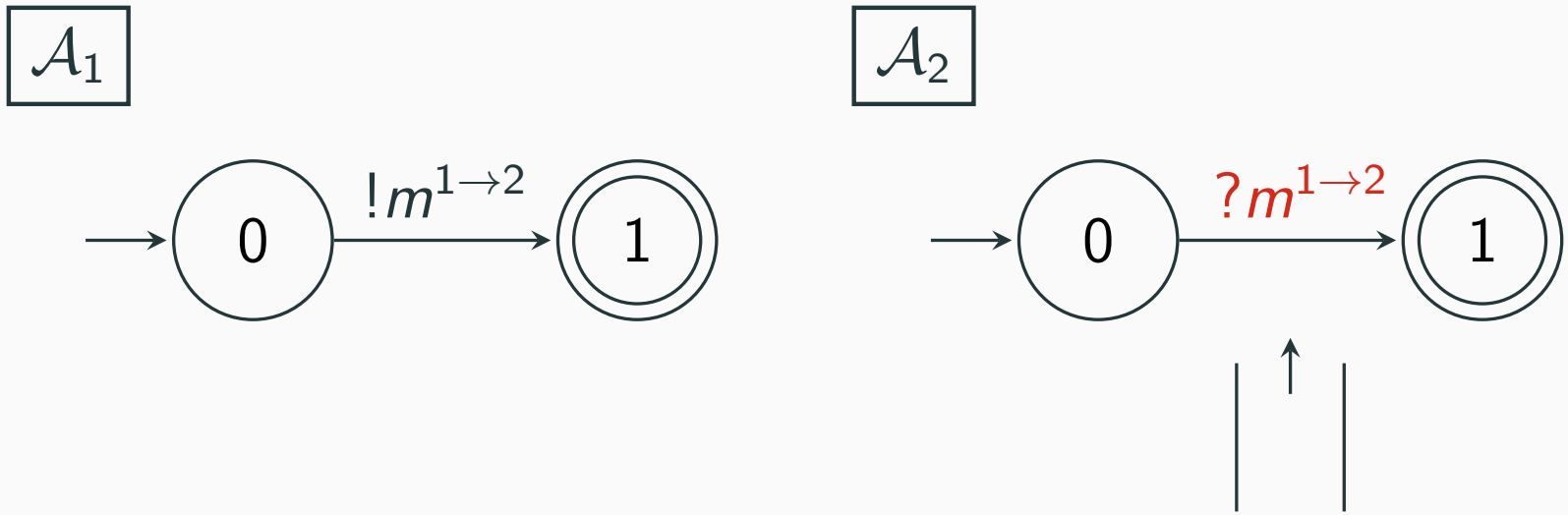
Communicating Automata

Finite state automata enriched with buffers to store exchanged messages



Communicating Automata

Finite state automata enriched with buffers to store exchanged messages



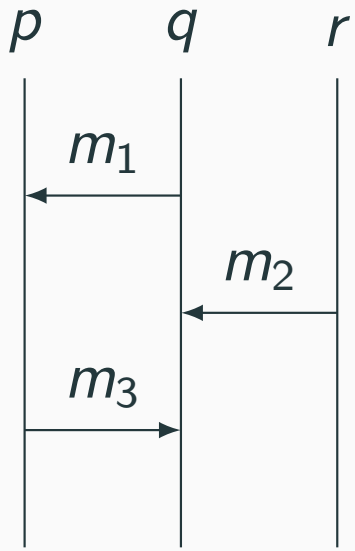
Message Sequence Chart (MSC)

Graphical representation of executions

- MSCs focus on the partial order among actions

  $p : ?m_1 < !m_3$ et

  $q : !m_1 < ?m_2 < ?m_3$



- Possible linearizations:

  $e_1 = !m_1\ ?m_1\ !m_2\ ?m_2\ !m_3\ ?m_3$

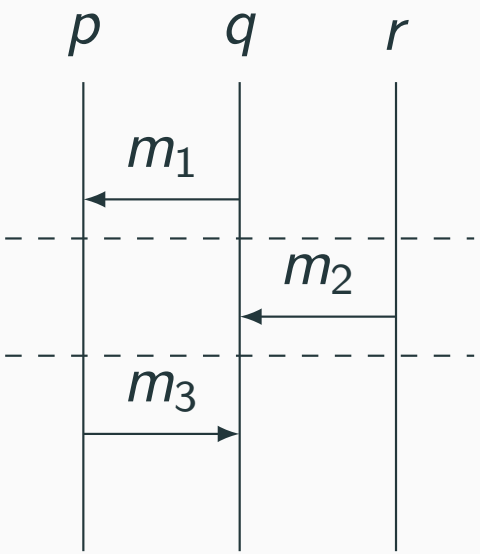
  $e_2 = !m_1\ !m_2\ ?m_1\ !m_3\ ?m_2\ ?m_3$

  $msc(e_1) = msc(e_2)$

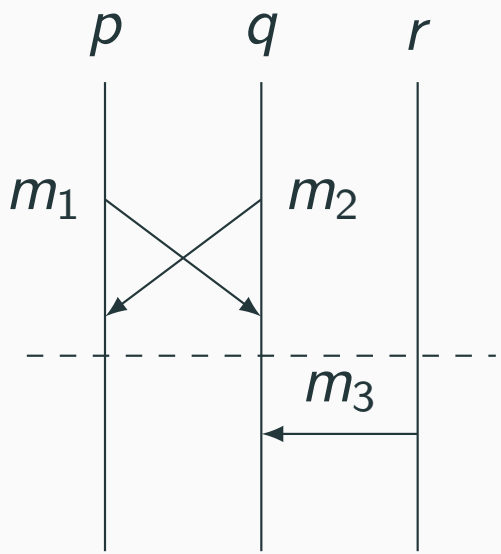
k-Synchronizability

k-Exchange

- k send actions followed by at most k receptions



1-exchanges



2-exchanges

- A k-exchange is an exchange

A k-synchronizable system

Causal delivery

Receptions match the order of send actions:

$$\text{if } !m_1 <!m_2 \text{ then } ?m_1 <?m_2$$

k-synchronous MSC

1. There is a linearization that satisfies causal delivery
2. The MSC is divisible into k-exchanges

k-synchronizable system

A system is **k-synchronizable** if all its executions are k-synchronizable.

A k-synchronizable system

Causal delivery

Receptions match the order of send actions:

$$\text{if } !m_1 <!m_2 \text{ then } ?m_1 <?m_2$$

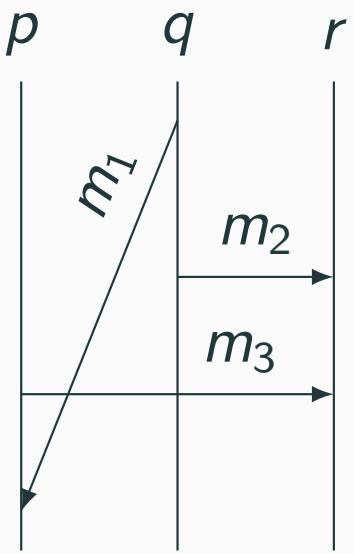
k-synchronous MSC

1. There is a linearization that satisfies causal delivery
2. The MSC is divisible into k-exchanges

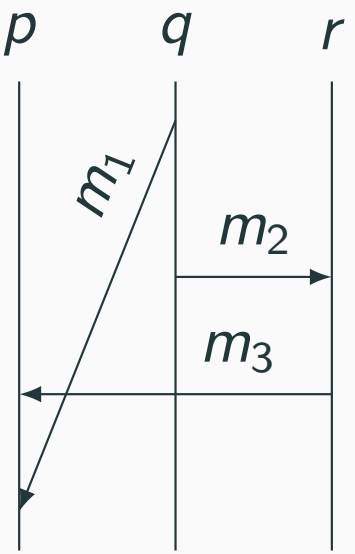
k-synchronizable system

A system is **k-synchronizable** if all its executions are k-synchronizable.

A linearization that satisfies causal delivery

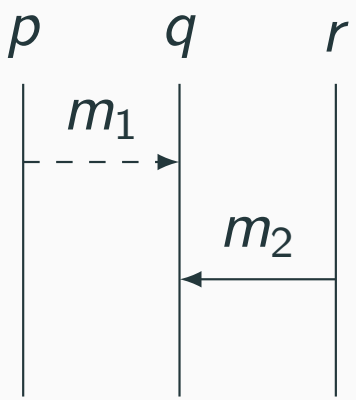


$!m_3 <?m_1$ and $!m_1 <!m_2$
and $?m_2 <?m_3$
$\rightarrow !m_1!m_2?m_2!m_3?m_3?m_1$



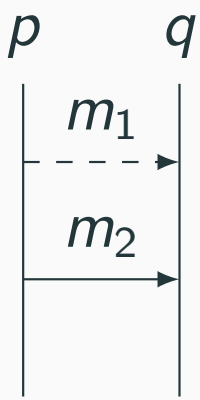
$?m_3 <?m_1$ and $!m_1 <!m_2$
and $?m_2 <!m_3$
$\rightarrow \underline{!m_1}!m_2?m_2\underline{!m_3}?m_3?m_1$

A linearization that satisfies causal delivery



No constraints

$\rightarrow !m_2 !m_1 ?m_2$



$!m1 < !m2$

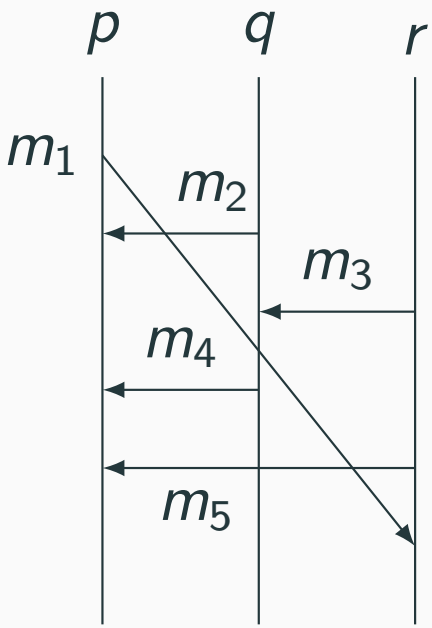
$\rightarrow !m_1 !m_2 ?m_2$

A k-synchronous MSC

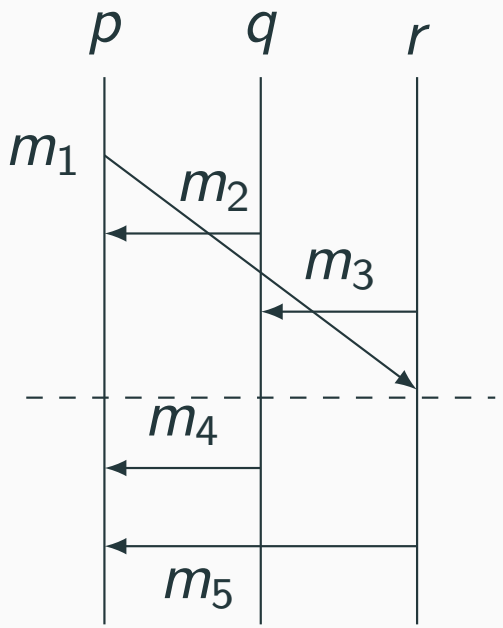
k-synchronous MSC

1. There is a linearization that satisfies causal delivery
2. The MSC is divisible into k-exchanges

The MSC is divisible into k-exchanges



Not divisible in k-exchanges



Divisible in k-exchanges

State of the art

- For a given k, is this system k-synchronizable ?

  $\Rightarrow$ **Decidable**

  - Proven in [Bouajjani *et al.*, 2018]
  - Adjusted and adapted to peer-to-peer systems in [Di Giusto *et al.*, 2020]

- Global idea
  - Looking for an execution not k-synchronizable

- Limitation: we need to set k

And now?

New problem !

Can we compute k such that a system is k-synchronizable ?

Yes!

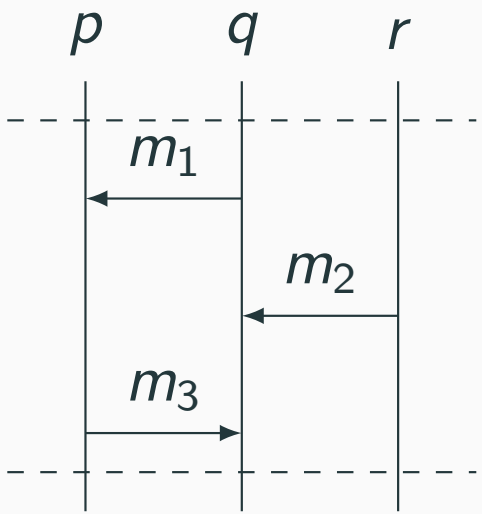
$\rightarrow$ Today: How to do that?

Characterization of k

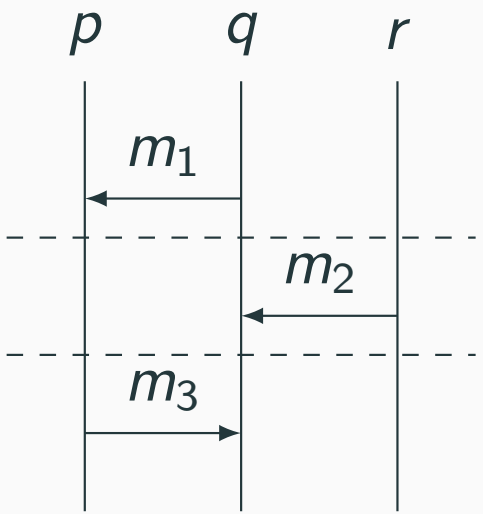
Prime exchange and Reachable exchange

Definition : Prime exchange

An exchange μ is **prime** if there is no decomposition $\mu = \mu_1 \cdots \mu_n$.



A not prime exchange

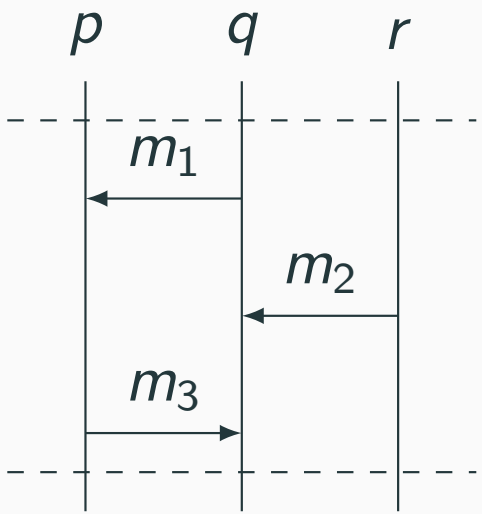


3 prime exchanges

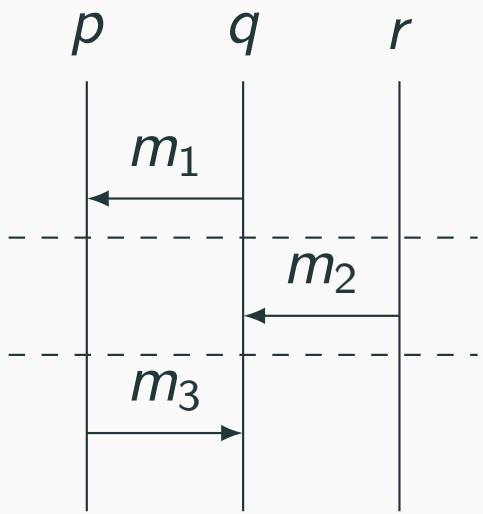
Prime exchange and Reachable exchange

Definition : Prime exchange

An exchange μ is **prime** if there is no decomposition $\mu = \mu_1 \cdots \mu_n$.



A not prime exchange



3 prime exchanges

Definition : Reachable exchange

An exchange μ is **reachable** if there are n exchanges $\mu_1 \cdots \mu_n$ such that $\mu_1 \cdots \mu_n \cdot \mu$ is an MSC of the system.

Characterization of k

Observation

A system is k-synchronizable $\Rightarrow$ all exchanges are $\leq k$.

k corresponds to
the size of the largest **prime** and **reachable** exchange

To find k:

- Search for all exchanges
- Take the largest one
- Compute its size

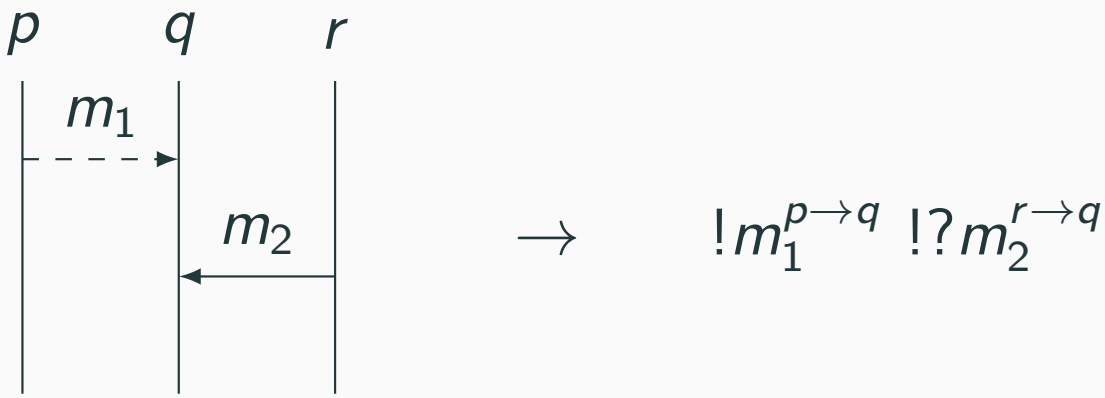
Searching exchanges

How to search for all exchanges ?

1. Encode exchanges by words
2. Show that the set of reachable exchanges is a regular effective language
3. Show that the set of prime exchanges is a regular effective language
4. Find the size of the largest word in the intersection

Encoding exchanges

A partial order $\rightarrow$ a word of matched and unmatched messages



Searching exchanges

How to search for all exchanges ?

1. Encode exchanges by words
2. Show that the set of reachable exchanges is a regular effective language
3. Show that the set of prime exchanges is a regular effective language
4. Find the size of the largest word in the intersection

Reachable exchanges

- We want to show that the language of reachable exchanges is regular
- 3 steps :
  - an automaton from global states
  - an automaton to verify causal delivery
  - a combination of both

Reachable exchanges - Automaton of global states

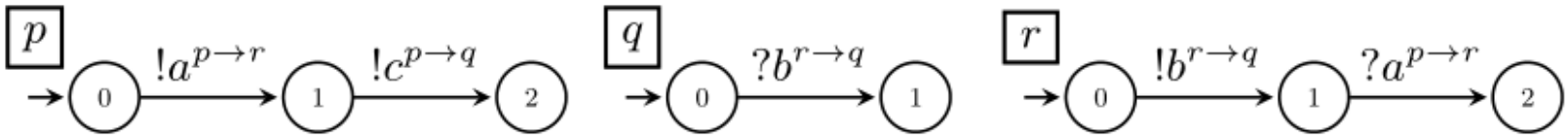
An exchange

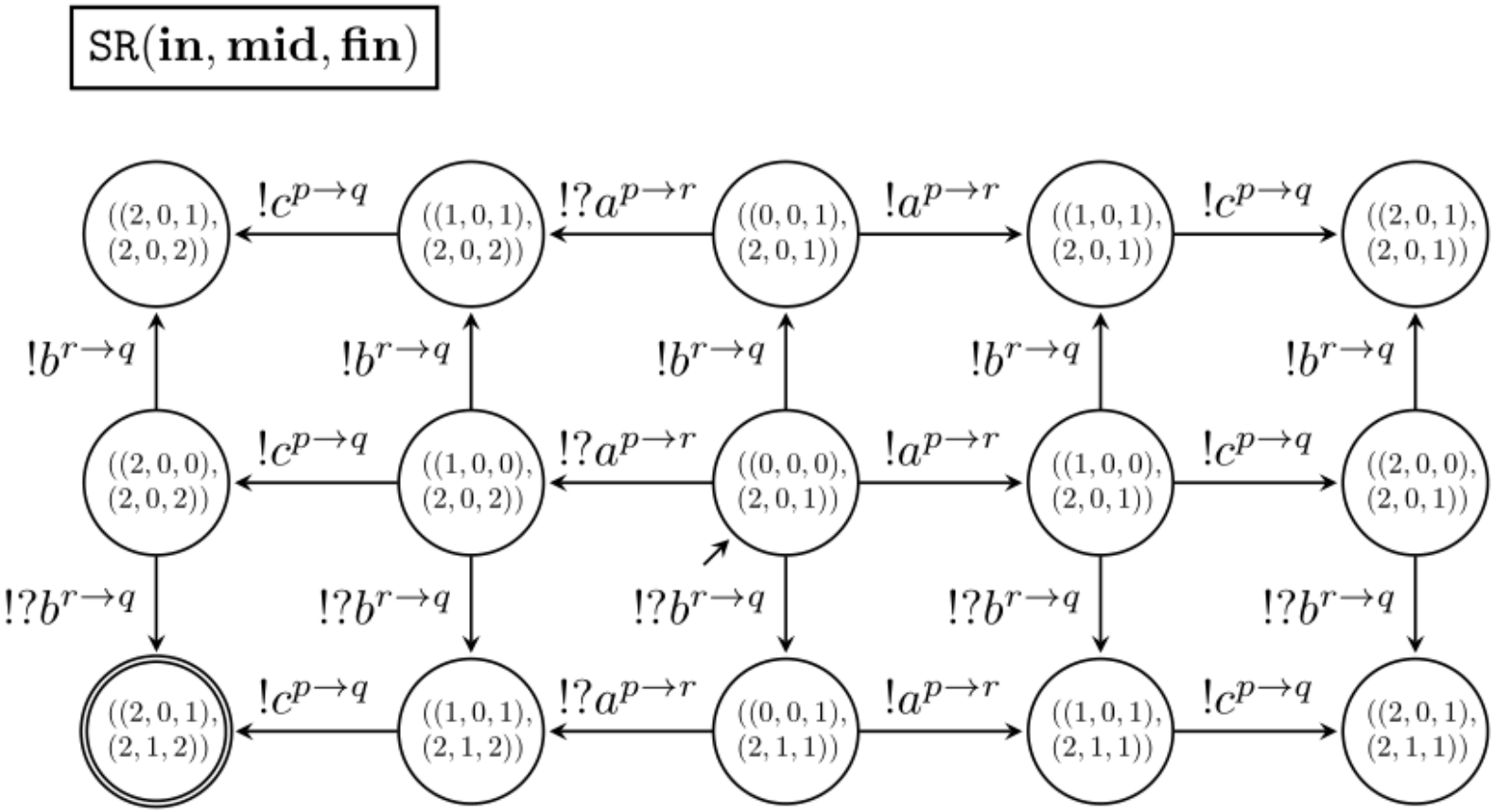
1. starts from a global state (*in*)
2. does only sends
3. transits through a middle state (*mid*)
4. does only receptions
5. arrives in a final state (*fin*)

Building: sort of product of sends from *init* to *mid* and of receptions from *mid* to *fin*

Reachable exchanges - Automaton of global states

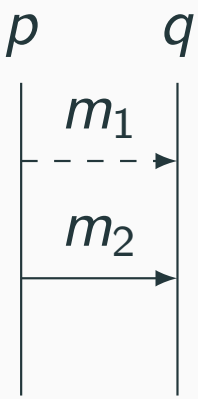
$in = (0,0,0), mid = (2,0,1), fin = (2,1,2)$





Reachable exchanges - Causal delivery automaton

Idea $\rightarrow$ store who have send unmatched messages



p is not allowed to send messages to q

Language of reachable exchanges

1. Language of causal delivery exchanges depends on
   - a triple of global state and
   - an initial and a final state of buffers

2. Concatenate them to obtain the **regular** language of reachable exchanges

Searching exchanges

How to search for all exchanges ?

1. Encode exchanges by words
2. Show that the set of reachable exchanges is a regular effective language
3. Show that the set of prime exchanges is a regular effective language
4. Find the size of the largest word in the intersection

Prime exchanges

Prime

An exchange μ is **prime** if there is no decomposition $\mu = \mu_1 \cdots \mu_n$.

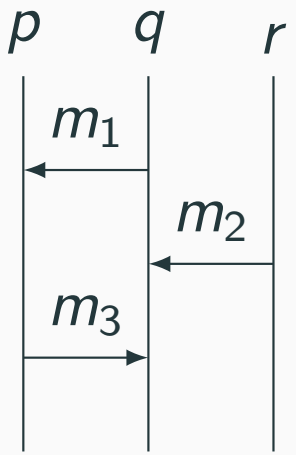
How to determine if an exchange is prime ?

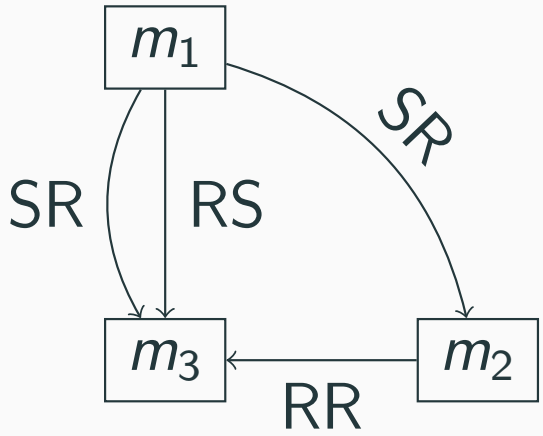
$\Rightarrow$ With its conflict graph

Conflict Graph

- It shows messages and their dependencies
- Labels express the type of dependency

  (S = Send, R = Receive)





- 1 SCC $\rightarrow$ prime exchange
- more than 1 SCC $\rightarrow$ not prime exchange

Prime exchange - An abstract conflict graph

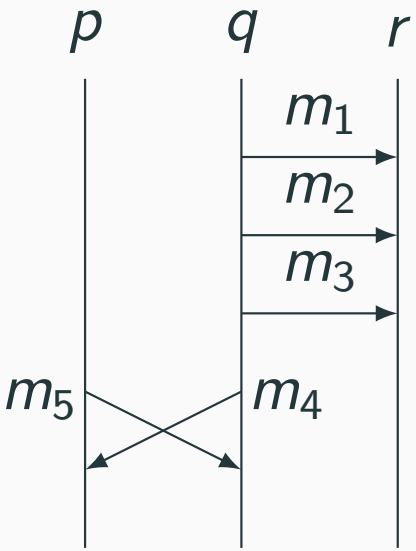
But build all possible conflict graphs : not possible

$\Rightarrow$ Abstraction !

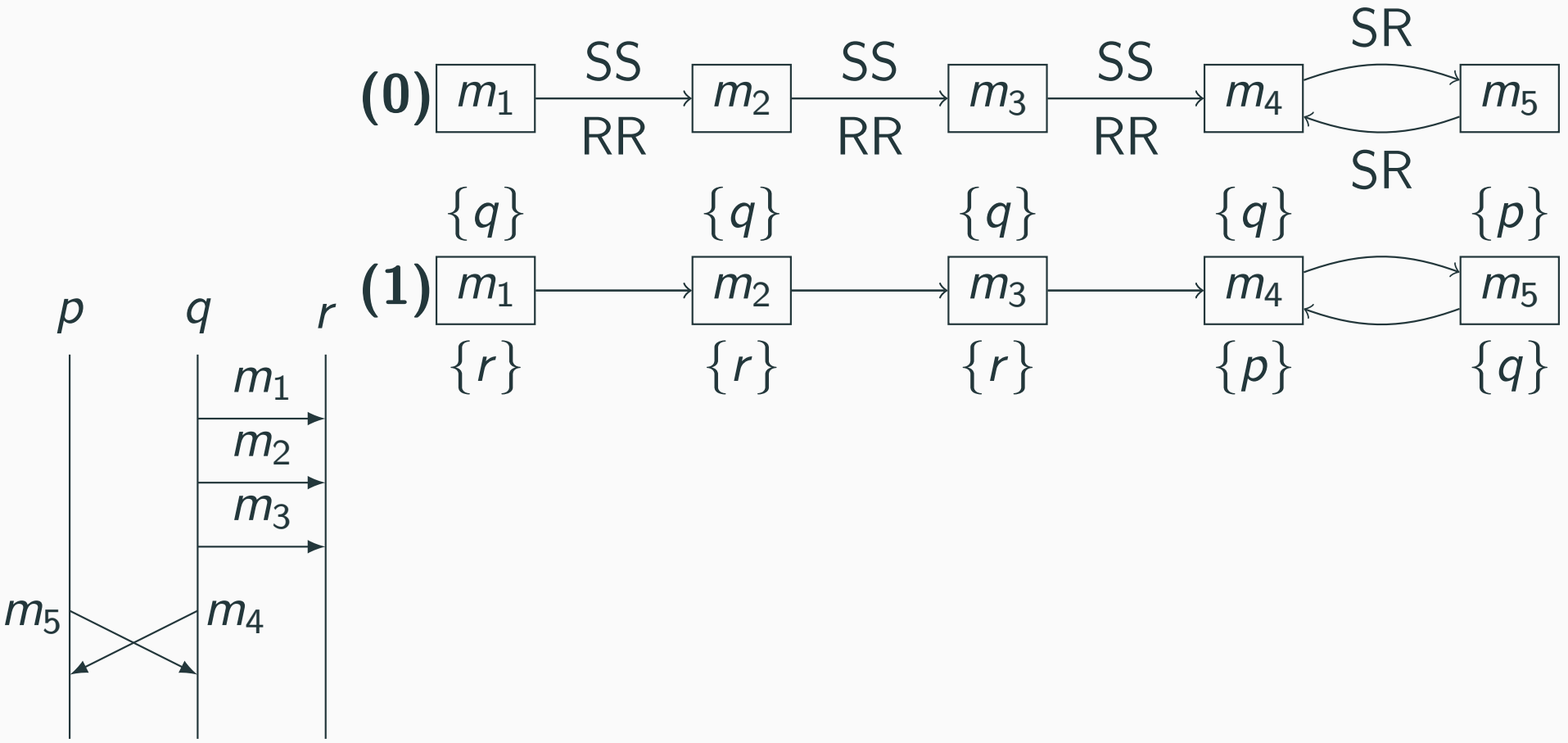
In 4 steps :

1. Add processes
2. Merging nodes
3. Delete useless processes
4. Delete useless nodes

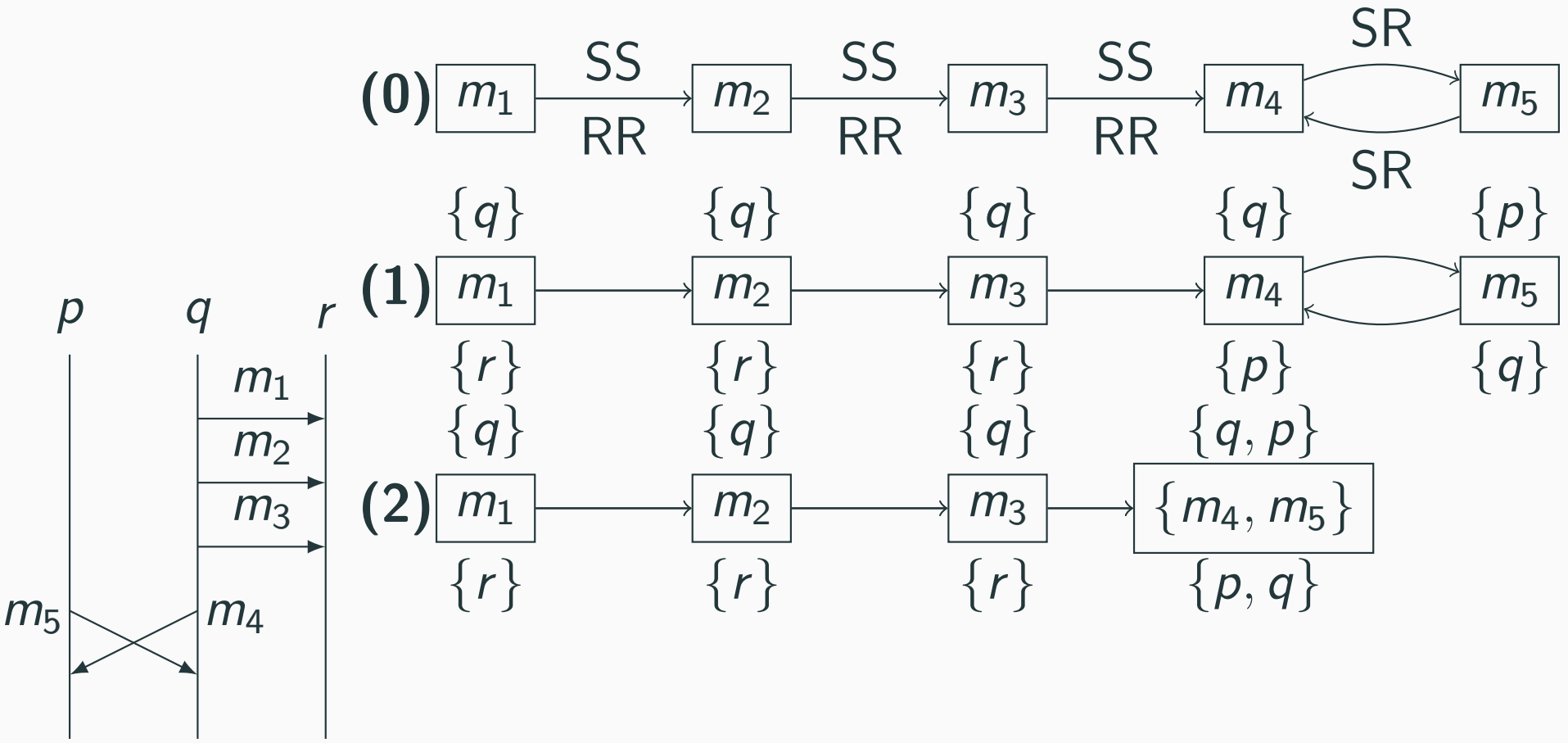
Prime exchange - An abstract conflict graph



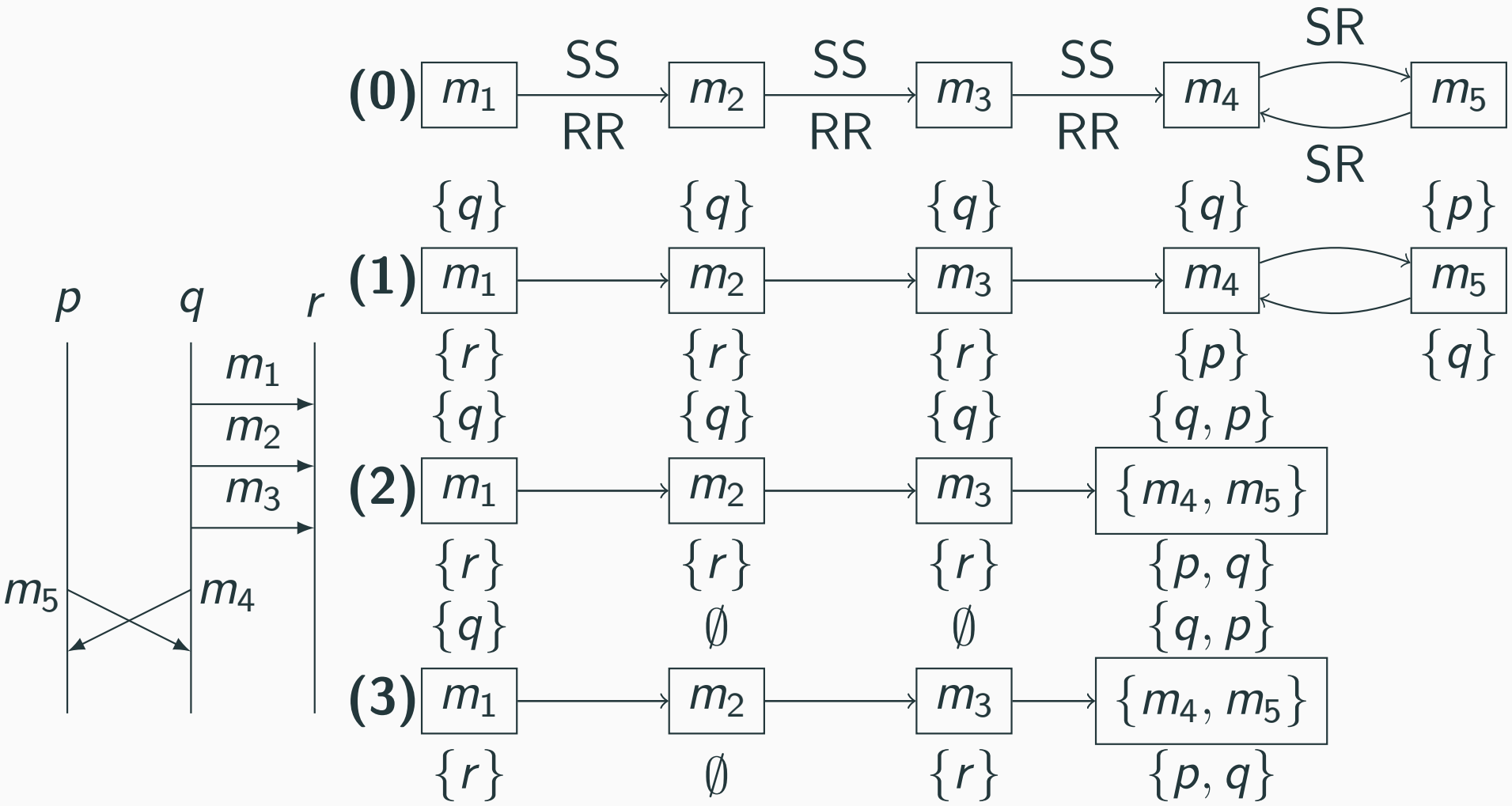
Prime exchange - An abstract conflict graph



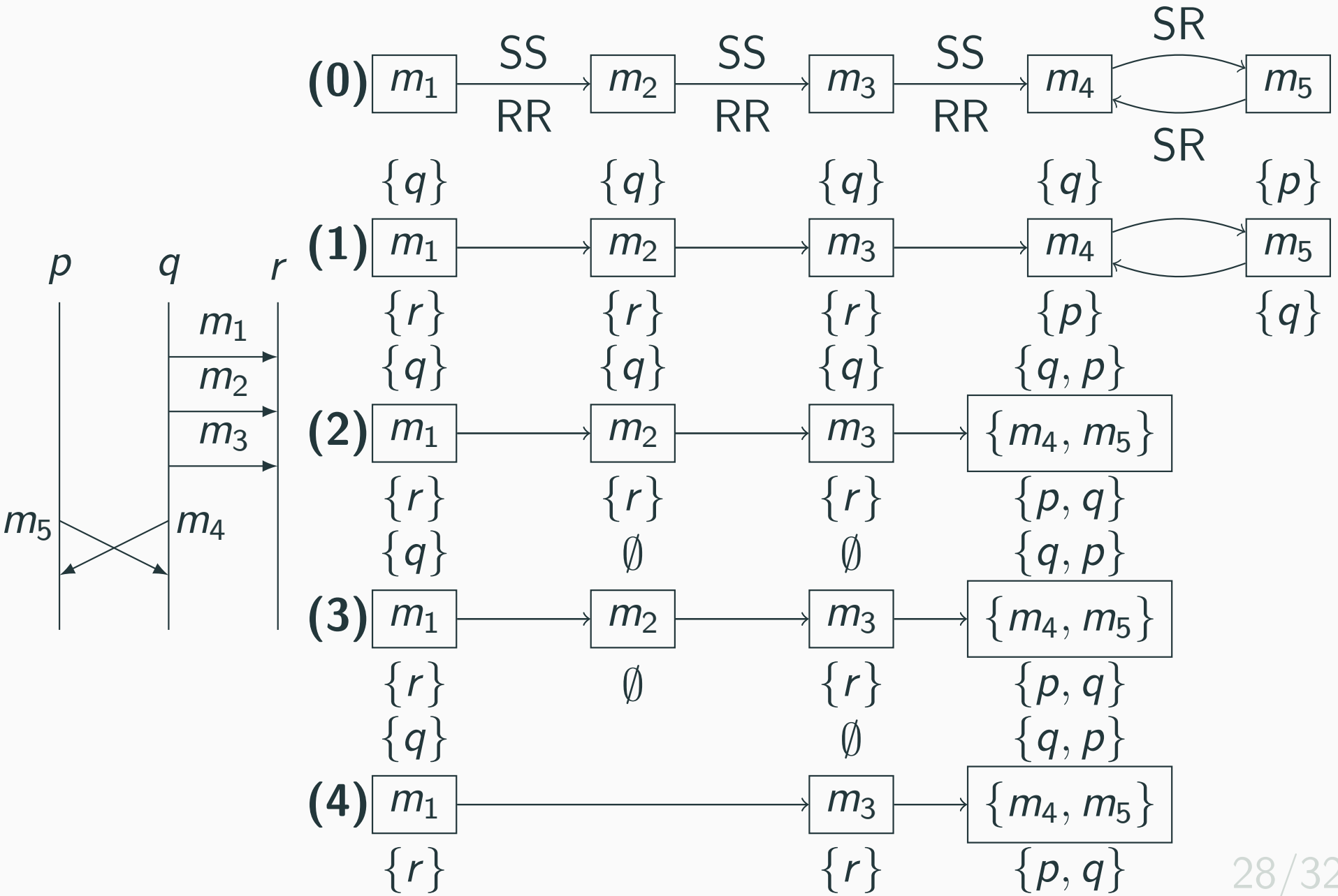
Prime exchange - An abstract conflict graph



Prime exchange - An abstract conflict graph



Prime exchange - An abstract conflict graph



Prime exchange - Automaton of conflict graphs

- Each state is an abstraction of conflict graph
- Each transition add a message
- Final states $\Leftrightarrow$ Conflict graphs with 1 SCC

$\Rightarrow$ The language of prime exchanges is **regular**

Searching exchanges

How to search for all exchanges ?

1. Encode exchanges by words
2. Show that the set of reachable exchanges is a regular effective language
3. Show that the set of prime exchanges is a regular effective language
4. Find the size of the largest word in the intersection

Computation of k

Computation of k

We are searching for the *length of the largest prime reachable exchange*

We have seen that

- language of reachable exchanges is regular
- language of prime exchanges is regular

$\Rightarrow$ Then we can find the largest prime reachable exchange

Last things:

- We need to test it
- If the system is not k-synchronizable, there is no other k' such that the system is k'-synchronizable

Conclusion and future works

- Conclusion
  - We knew how to test k-synchronizability for a give k
  - Now, we can guess the k

- Future works : extend definition of k-synchronizability

Thank you!

Bibliography

- Di Giusto, Laversa & Lozes. Guessing the buffer bound for k-synchronizability. To be published in *CIAA* 2021
- Di Giusto, Laversa & Lozes. On the k-synchronizability of Systems. In *Fossacs* 2020, 157-176.
- Lohrey & Muscholl. Bounded MSC communication. *Information and Computation*, 2004, 189(2), 160-181.
- Basu & Bultan. On deciding synchronizability for asynchronously communicating systems. *Theoretical Computer Science*, 2016, 656, 60-75.
- Bouajjani, Enea, Ji & Qadeer. On the completeness of verifying message passing programs under bounded asynchrony. In *Cav* 2018 372-391.

When a system is not k-synchronizable

$\Rightarrow$ A system is not k-synchronizable iff there exists a borderline violation

Borderline violation

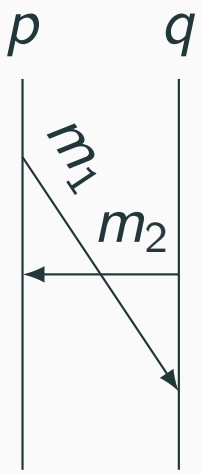
It is an execution that is not k-synchronizable but it can be made by removing the last reception.

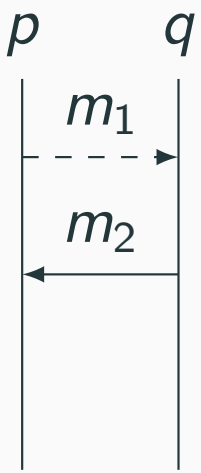
Borderline violation

Borderline violation

It is an execution that is not k-synchronizable but it can be made by removing the last reception.

$$\mathbf{e} = !m_1^{p\to q}\ !m_2^{q\to p}\ ?m_2^{q\to p}\ ?m_1^{p\to q} \quad \mathbf{e'} = !m_1^{p\to q}\ !m_2^{q\to p}\ ?m_2^{q\to p}$$

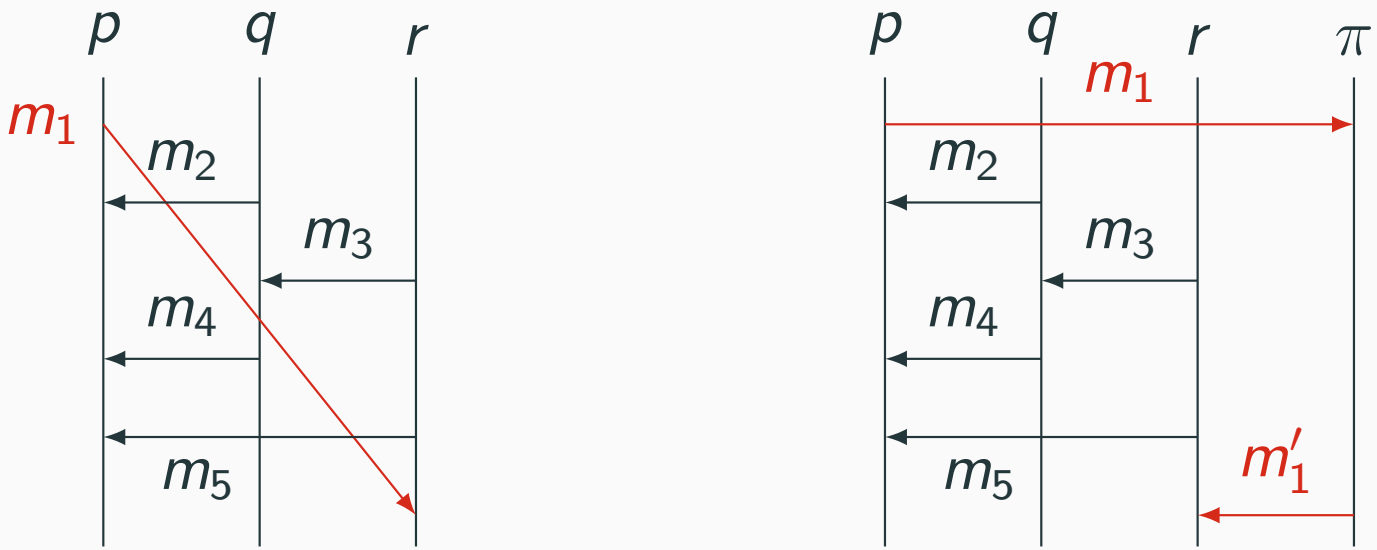




Is a system k-synchronizable? (for a given k)

The Trick

Construction of system with a deviated message



$\Rightarrow$ Borderline executions become k-synchronizable

The key observation

Feasible execution, bad execution

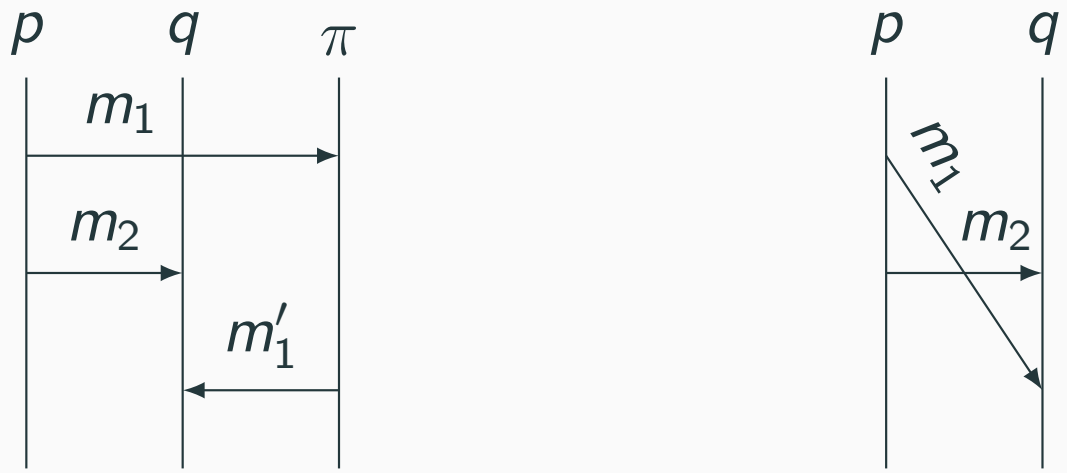
A k-synchronizable execution e is **feasible** if there is an execution $e' \cdot r$ such that $\mathit{deviate}(e' \cdot r) = e$.

An execution is **bad** if it is not k-synchronizable.

A system is not k-synchronizable iff there is a k-synchronizable deviated execution that is feasible and bad.

The Algorithm -1

1. Identify **causal delivery** executions in the deviated system

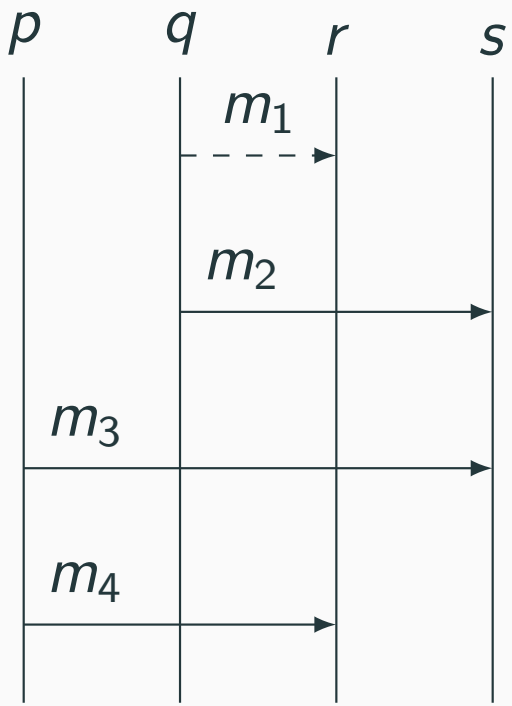


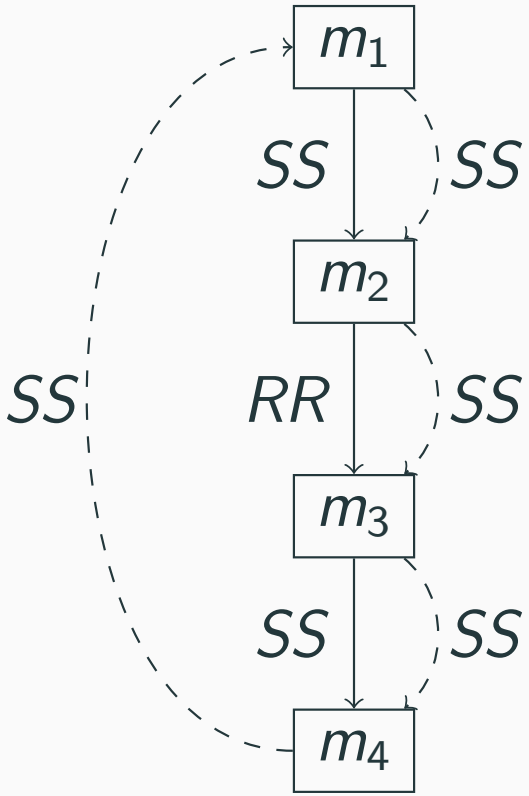
$\Rightarrow$ We select only **feasible** executions

Causal delivery

Characterization

An MSC satisfies causal delivery iff there is no cyclic dependency $m \overset{ss}{\dashrightarrow} m$ in its conflict graph.



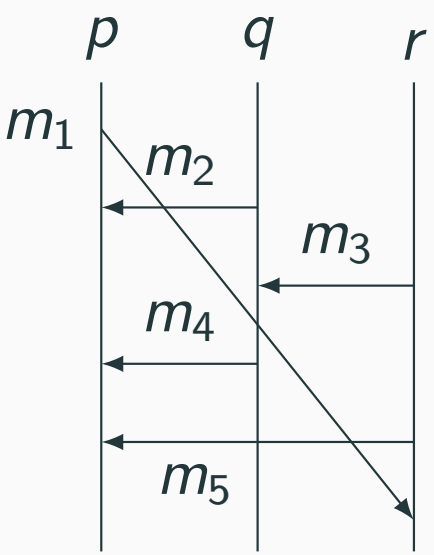


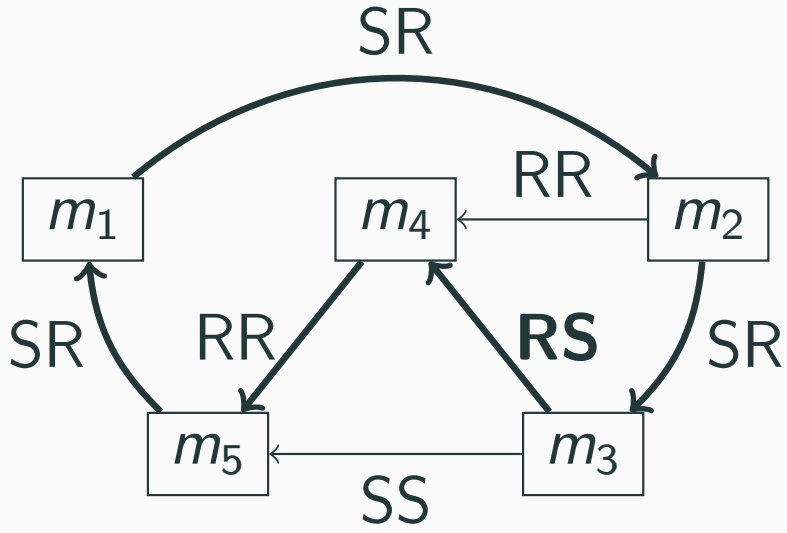
When an MSC is divisible into k-exchanges

Characterization

Its conflict graph does not contain any strongly connected component that:

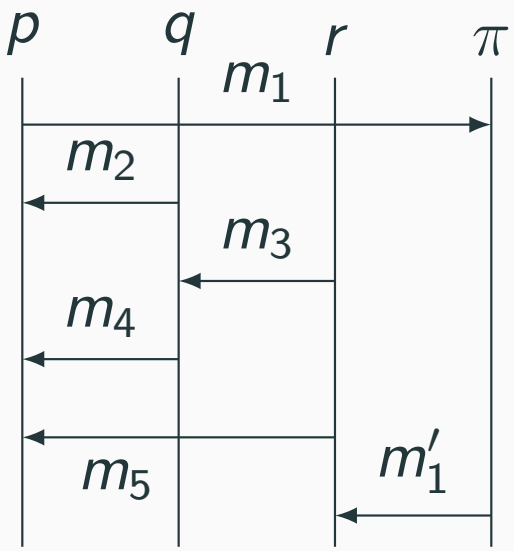
- is of size $> k$
- contains an RS label

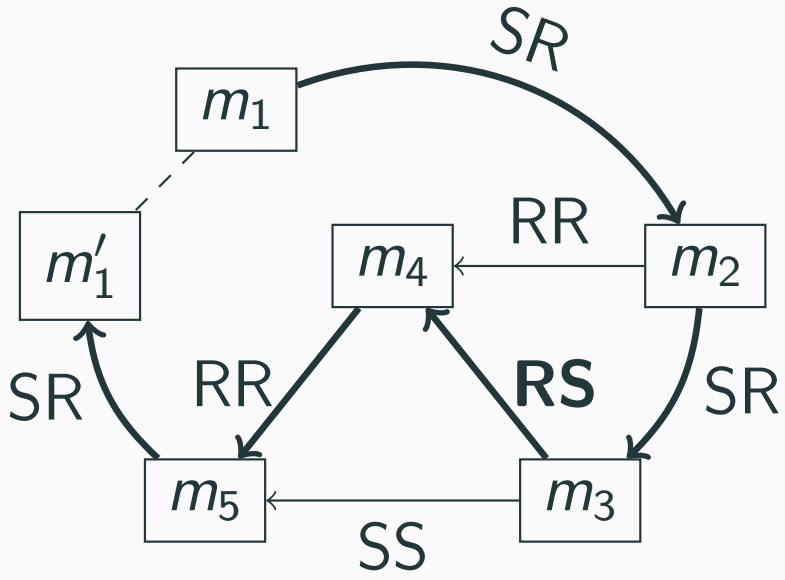




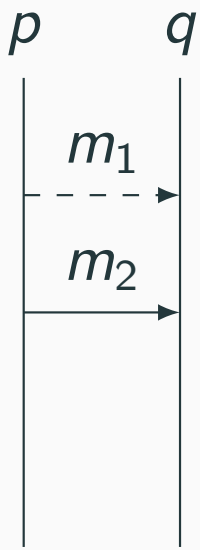
The Algorithm -2

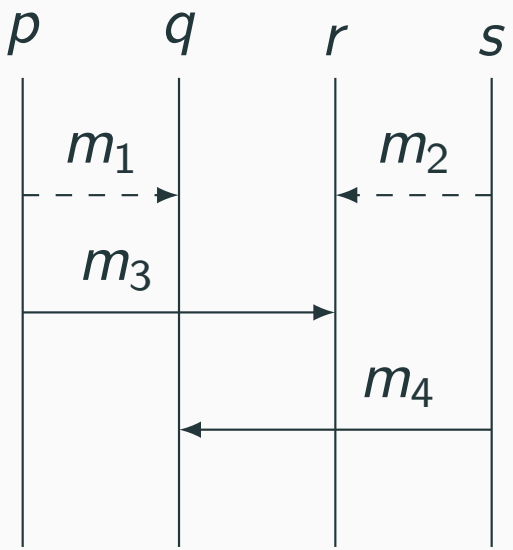
2. Recognize **bad** executions

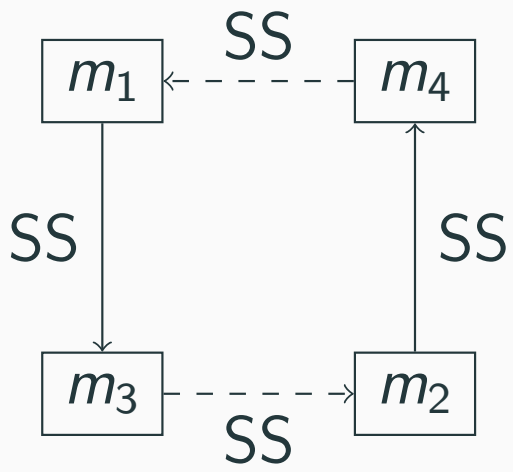




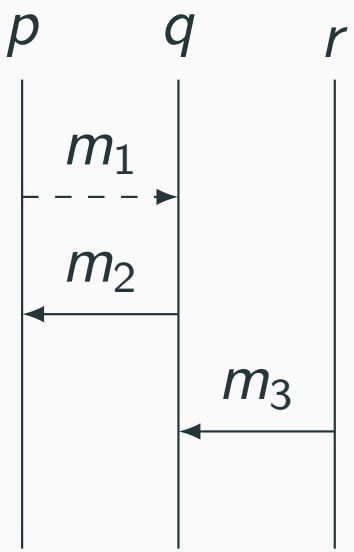
Causal delivery

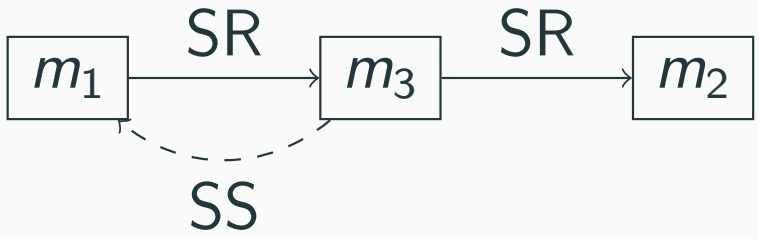






Example of the effect of unmatched messages

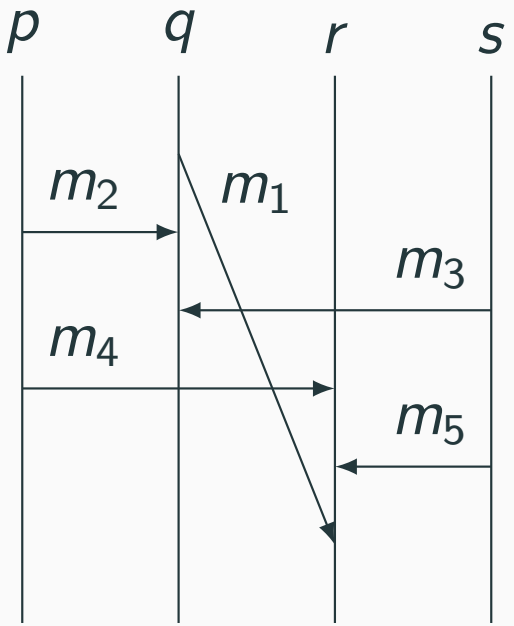


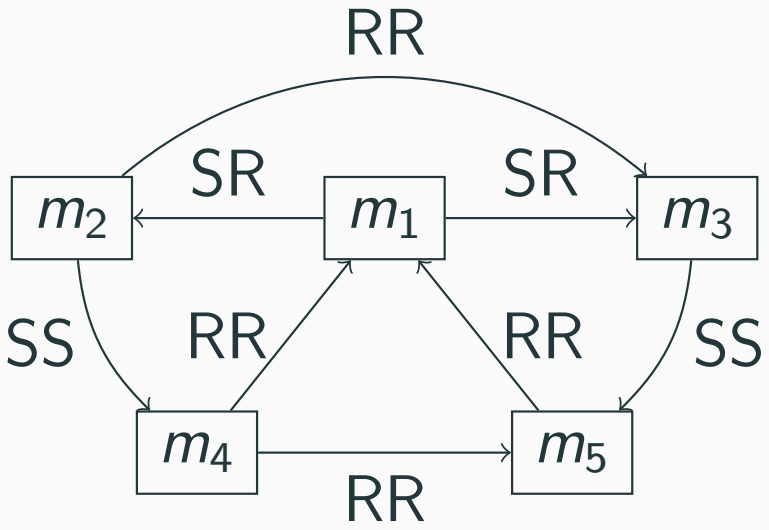


$$e = !m_3 !m_1 !m_2 ?m_3 ?m_2$$

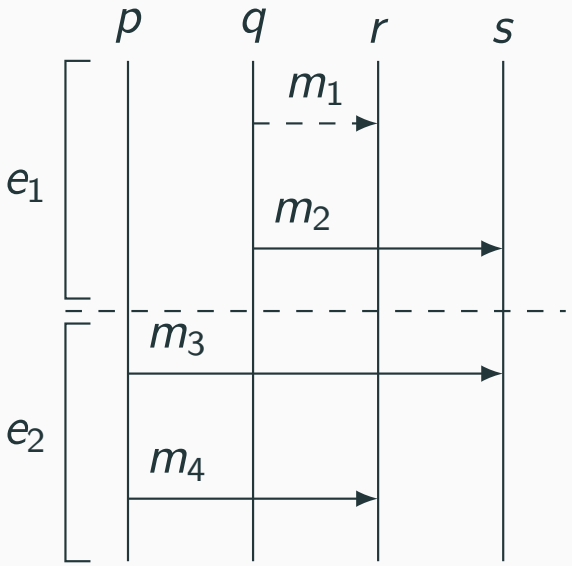
Counter-example of Bouajjani et al.

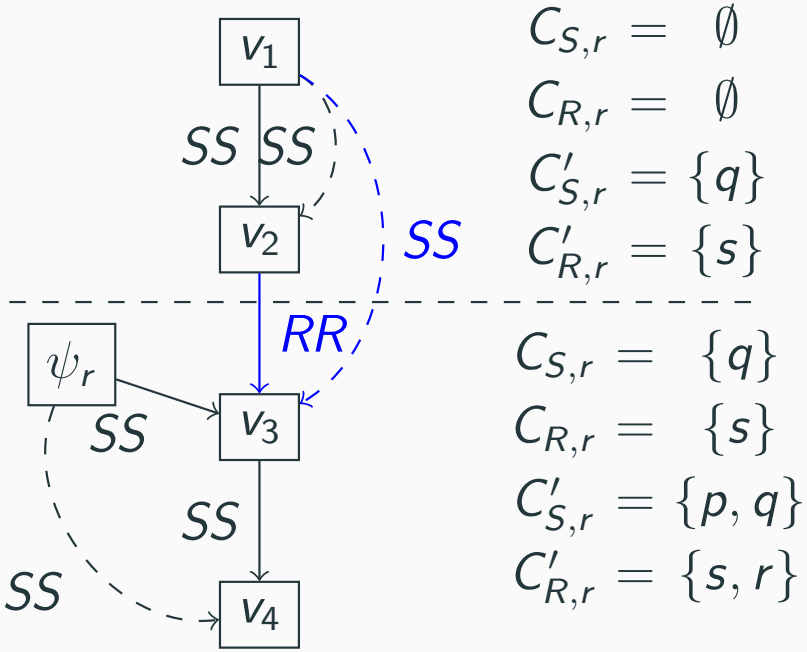
- The strongly connected component has size 5 while the longest cycle 4.





Extended Conflict Graphs of k-exchanges





$$C_{S,r} = \emptyset$$
$$C_{R,r} = \emptyset$$
$$C'_{S,r} = \{q\}$$
$$C'_{R,r} = \{s\}$$

$$C_{S,r} = \{q\}$$
$$C_{R,r} = \{s\}$$
$$C'_{S,r} = \{p, q\}$$
$$C'_{R,r} = \{s, r\}$$

How to find the original SCC of a deviated message