Local testing of MSC specifications

Madhavan Mukund

Chennai Mathematical Institute
http://www.cmi.ac.in/~madhavan

Joint work with Puneet Bhateja

DNTTT '08, Cremona, Italy
10 October 2008

Local testing of MSC specifications

Madhavan Mukund

Chennai Mathematical Institute
http://www.cmi.ac.in/~madhavan

Joint work with Puneet Bhateja
Extends joint work with Paul Gastin, K Narayan Kumar

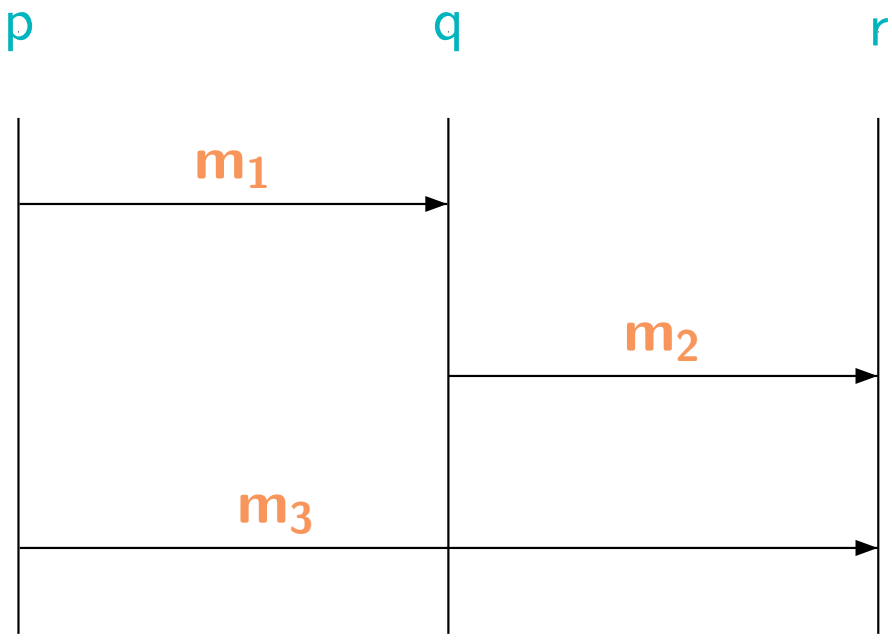
DNTTT '08, Cremona, Italy
10 October 2008

Scenarios

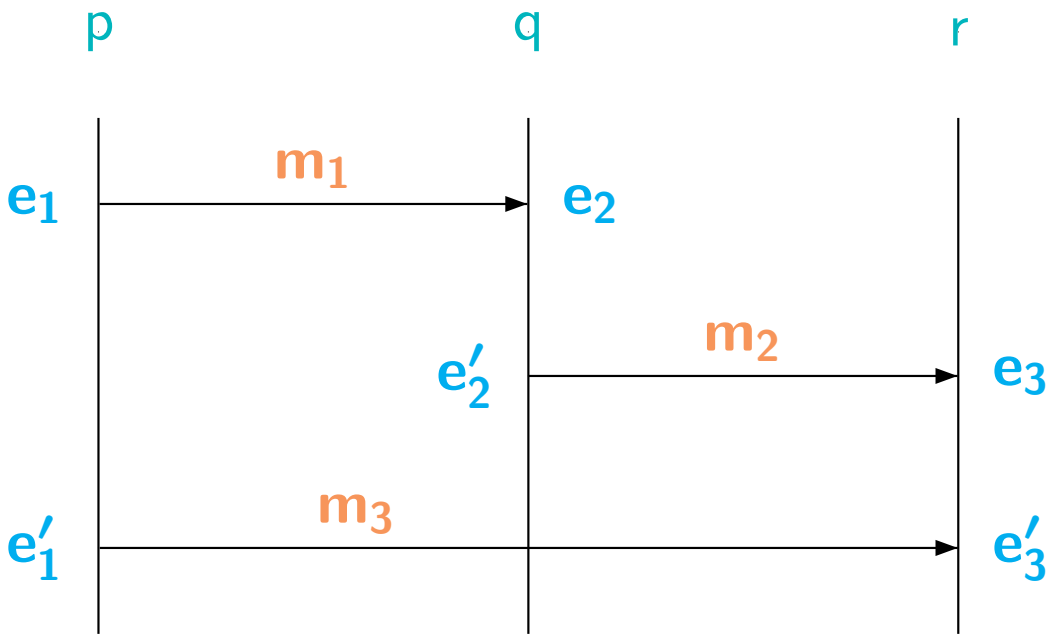
- Visual specification of message-passing system behaviour
- A scenario describes a pattern of interaction
- Telecommunications
  - Message sequence charts (MSC)
  - Messages sent between communicating agents
- UML
  - Sequence diagrams
  - Interaction between objects

    e.g., method invocations etc

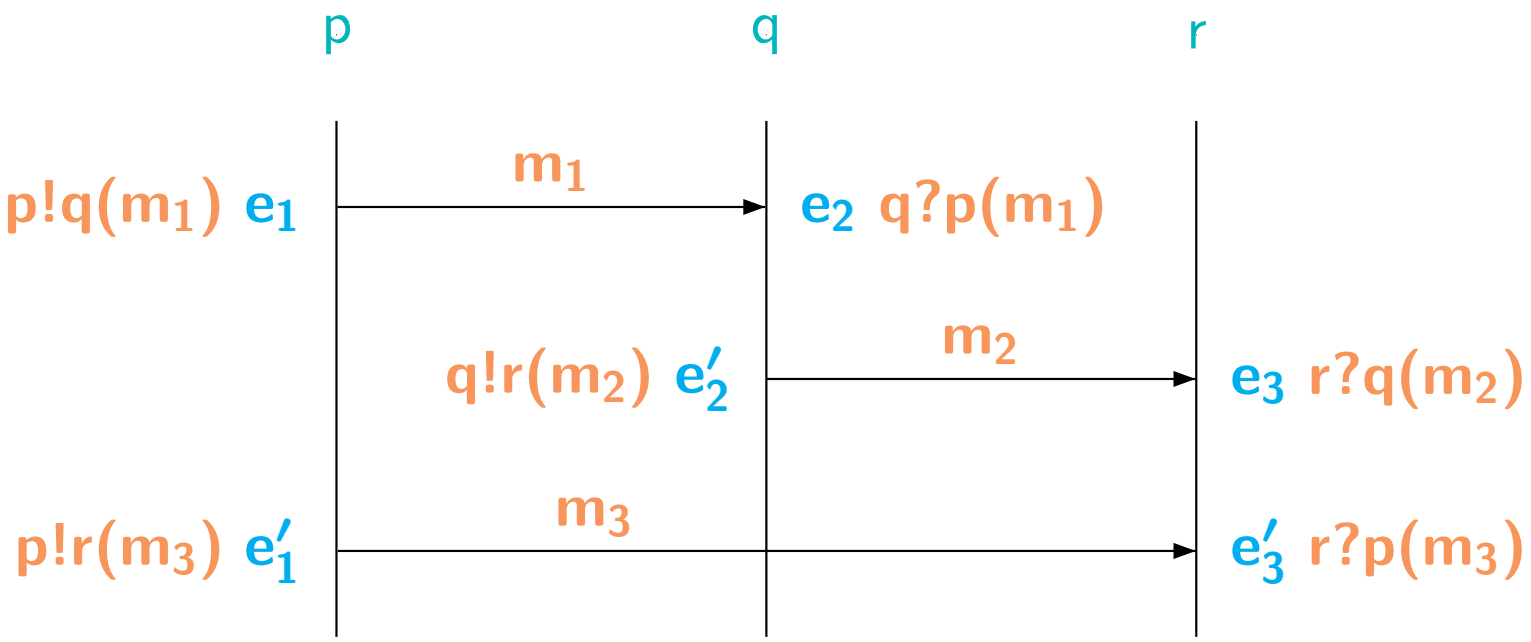
An MSC



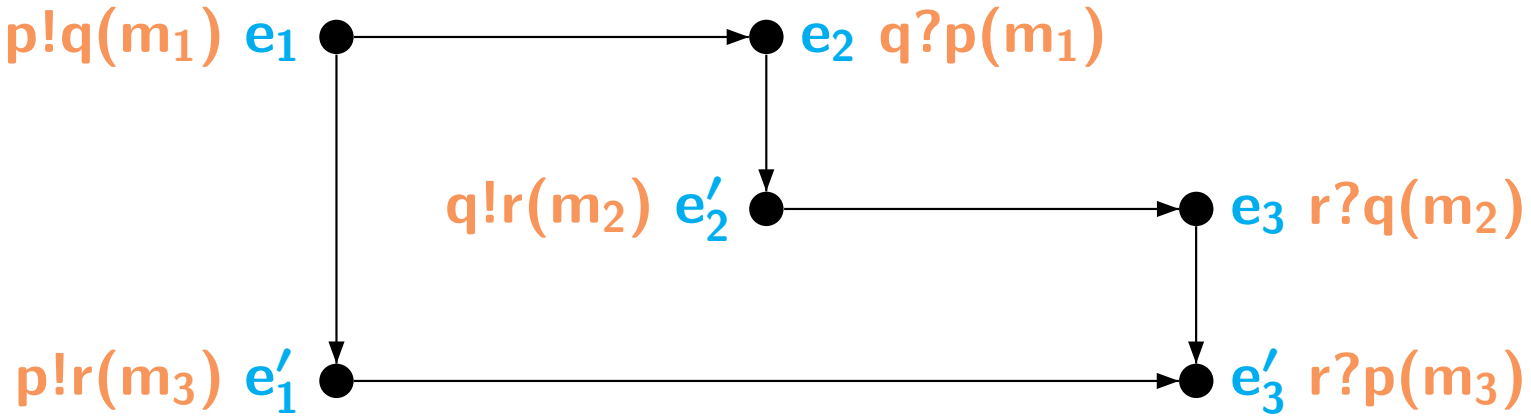
An MSC with events



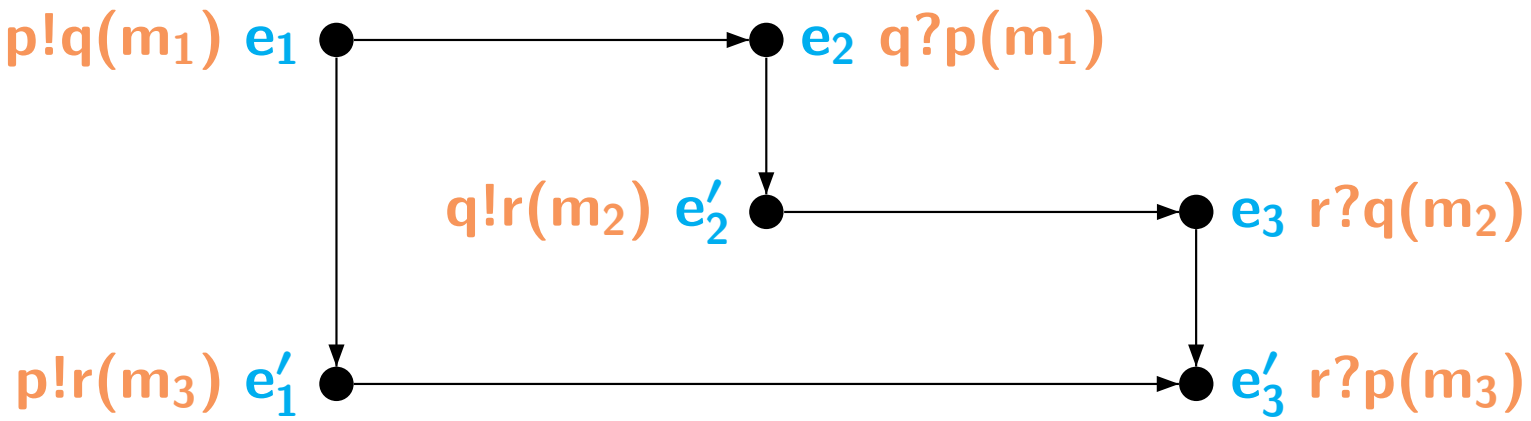
An MSC with events



MSCs as labelled partial orders



MSCs as labelled partial orders



- Linearizations give a word language
  $p!q(m_1)\ p!r(m_3)\ q?p(m_1)\ q!r(m_2)\ r?q(m_2)\ r?p(m_3)$,
  $p!q(m_1)\ q?p(m_1)\ q!r(m_2)\ p!r(m_3)\ r?q(m_2)\ r?p(m_3)$,...
- A single linearization is sufficient to reconstruct MSC

(Local) testing using scenarios

- Does an implementation conform to an MSC specification?
  - Are all observed behaviours legal?

(Local) testing using scenarios

- Does an implementation conform to an MSC specification?
  - Are all observed behaviours legal?
- Local testing
  - Selectively monitor the channels of the system
  - For each process **p**, local observer generates messages for **p** and records the responses
  - If each local observation is consistent with some MSC in the specification, the implementation passes the test

(Local) testing using scenarios

- Does an implementation conform to an MSC specification?
  - Are all observed behaviours legal?
- Local testing
  - Selectively monitor the channels of the system
  - For each process **p**, local observer generates messages for **p** and records the responses
  - If each local observation is consistent with some MSC in the specification, the implementation passes the test
- "Real world" testing—for example, using TTCN-3— is a version of local testing

(Local) testing using scenarios

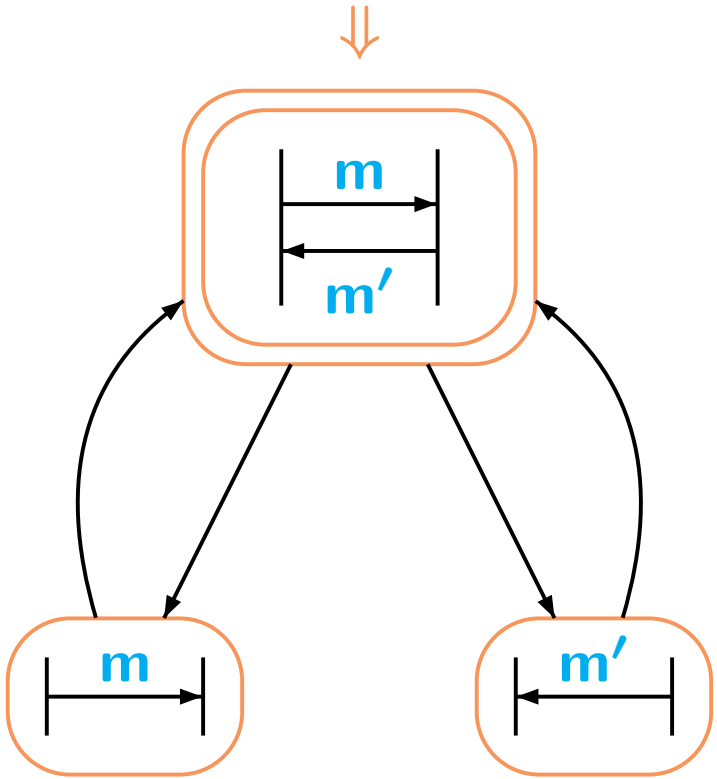
- Does an implementation conform to an MSC specification?
  - Are all observed behaviours legal?
- Local testing
  - Selectively monitor the channels of the system
  - For each process **p**, local observer generates messages for **p** and records the responses
  - If each local observation is consistent with some MSC in the specification, the implementation passes the test
- "Real world" testing—for example, using TTCN-3— is a version of local testing
- Does local testing suffice to check conformance for (regular) MSC specifications?

High level MSCs (HMSCs) to generate infinite collections

- A finite state automaton
- Each state is labelled by an MSC
- Each (legal) path in the automaton generates an MSC

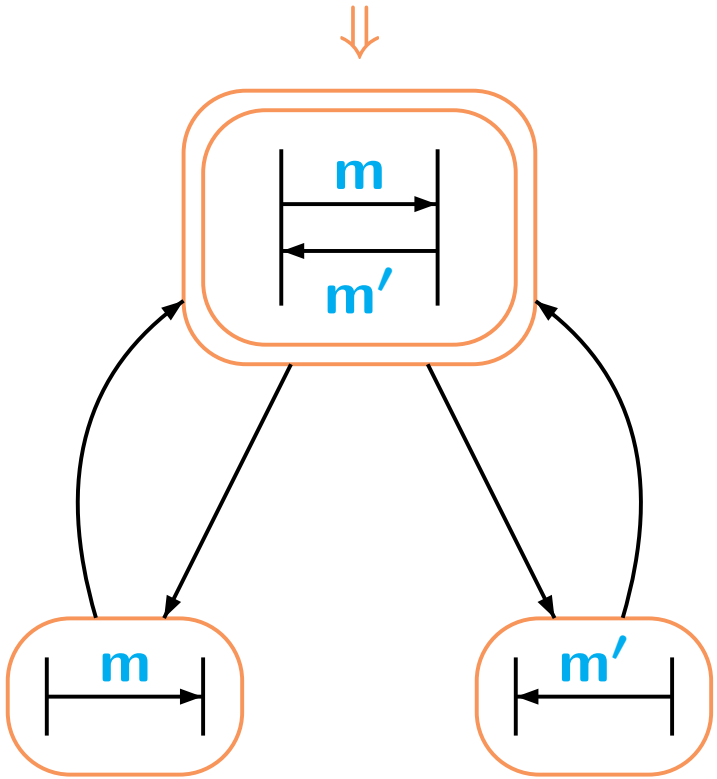
High level MSCs (HMSCs) to generate infinite collections

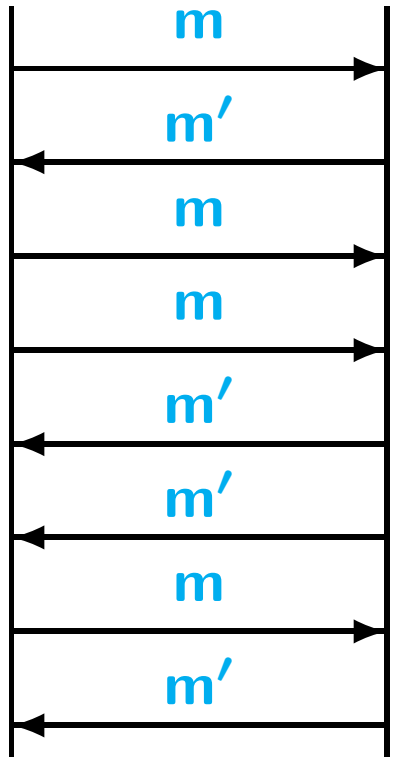
- A finite state automaton
- Each state is labelled by an MSC
- Each (legal) path in the automaton generates an MSC



High level MSCs (HMSCs) to generate infinite collections

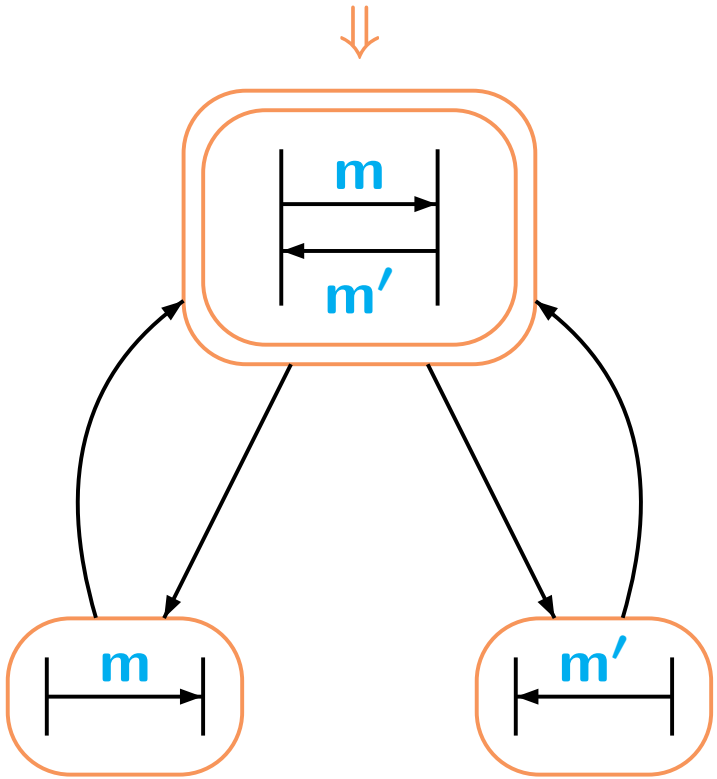
- A finite state automaton
- Each state is labelled by an MSC
- Each (legal) path in the automaton generates an MSC

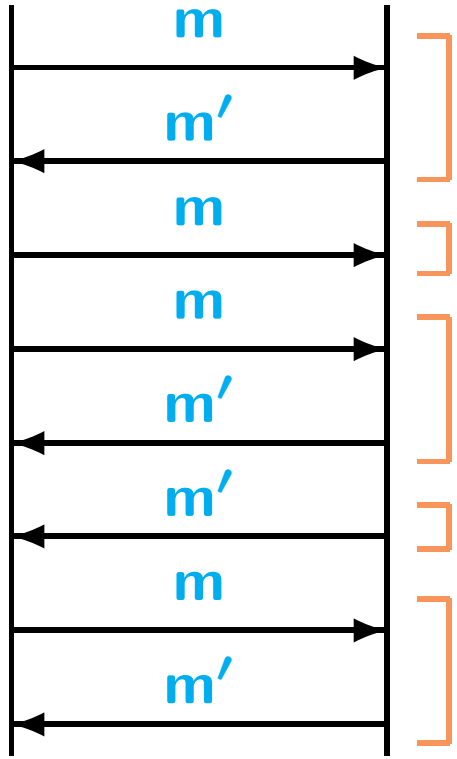




High level MSCs (HMSCs) to generate infinite collections

- A finite state automaton
- Each state is labelled by an MSC
- Each (legal) path in the automaton generates an MSC





HMSC semantics

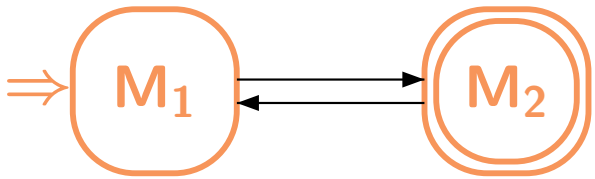
- All processes must traverse the same path in an HMSC

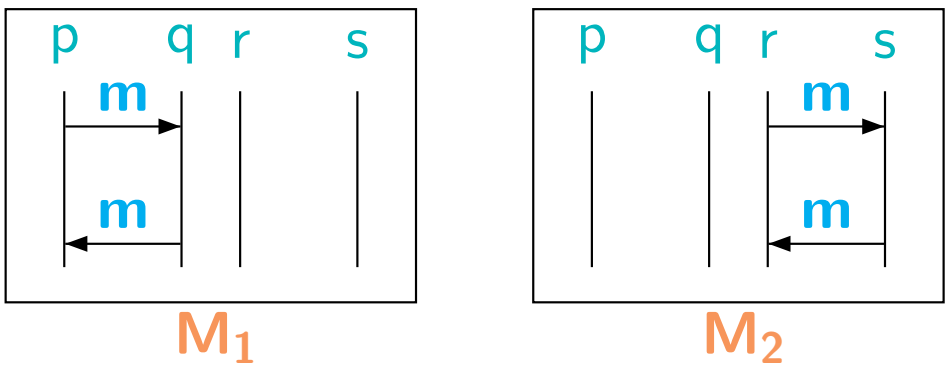
HMSC semantics

- All processes must traverse the same path in an HMSC
- . . . but processes move asynchronously
- Some processes may be (unboundedly) far ahead of others

HMSC semantics

- All processes must traverse the same path in an HMSC
- . . . but processes move asynchronously
- Some processes may be (unboundedly) far ahead of others





- After k iterations, we could have r and s in the final copy of M_2 while p and q are in the first copy of M_1

Regular MSC languages

- An MSC is (uniquely) determined by its linearizations
  - Set of strings over send actions **p!q(m)** and receive actions **p?q(m)**

- Collection of MSCs $\Leftrightarrow$
  word language over send/receive actions

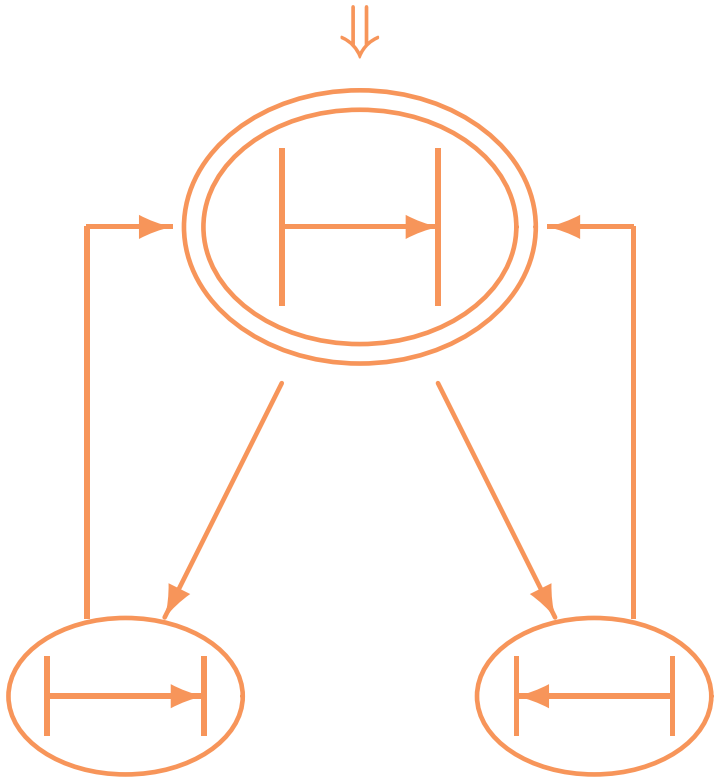
- Regular collection of MSCs $\stackrel{\triangle}{=}$
  linearizations form a regular language

HMSCs and regularity

- HMSC specifications may not be regular

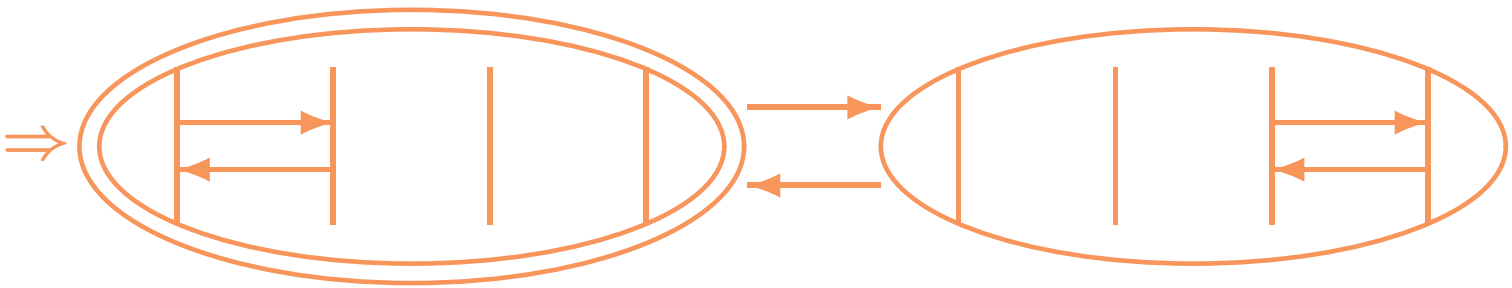
HMSCs and regularity

- HMSC specifications may not be regular
- Problem 1 Unbounded buffers



HMSCs and regularity

- HMSC specifications may not be regular
- Problem 1 Unbounded buffers
- Problem 2 Global synchronization yields context-free behaviours



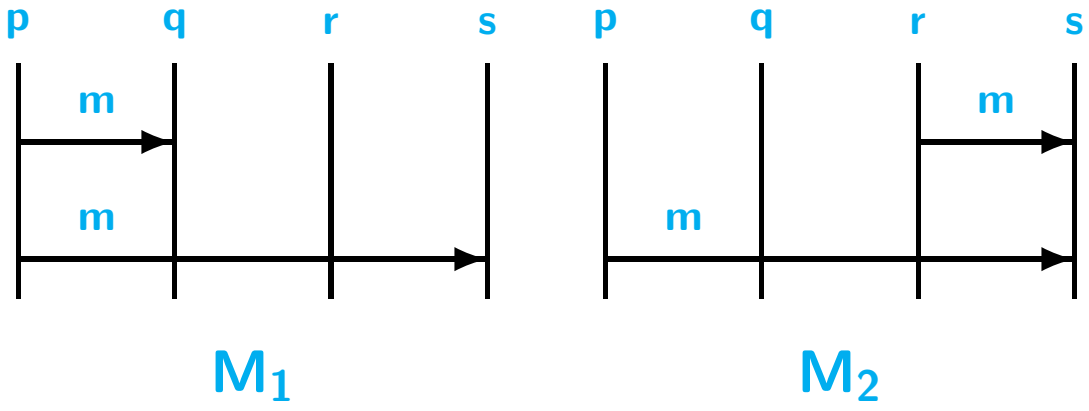
HMSCs and regularity

- HMSC specifications may not be regular
- Problem 1 Unbounded buffers
- Problem 2 Global synchronization yields context-free behaviours
- Sufficient structural conditions on HMSCs to guarantee regularity . . . [AY99,MP99]
  - Locally synchronized
- . . . but checking if an HMSC specification is regular is undecidable [HMNST05]

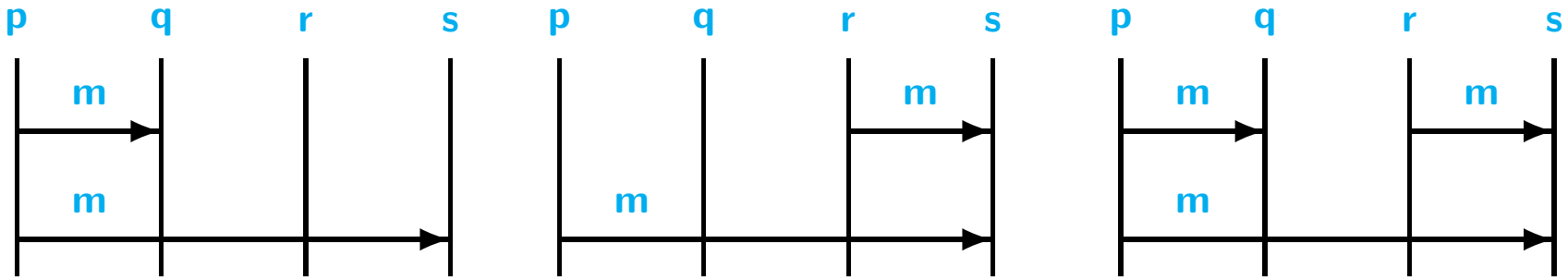
HMSCs and regularity

- HMSC specifications may not be regular
- Problem 1 Unbounded buffers
- Problem 2 Global synchronization yields context-free behaviours
- Sufficient structural conditions on HMSCs to guarantee regularity . . . [AY99,MP99]
  - Locally synchronized
- . . . but checking if an HMSC specification is regular is undecidable [HMNST05]
- Every regular MSC language can be implemented as network of communicating finite-state automata with (universally) bounded channels [HMNST05]
- This channel bound can be computed, if needed.

Implied scenarios [AEY00]



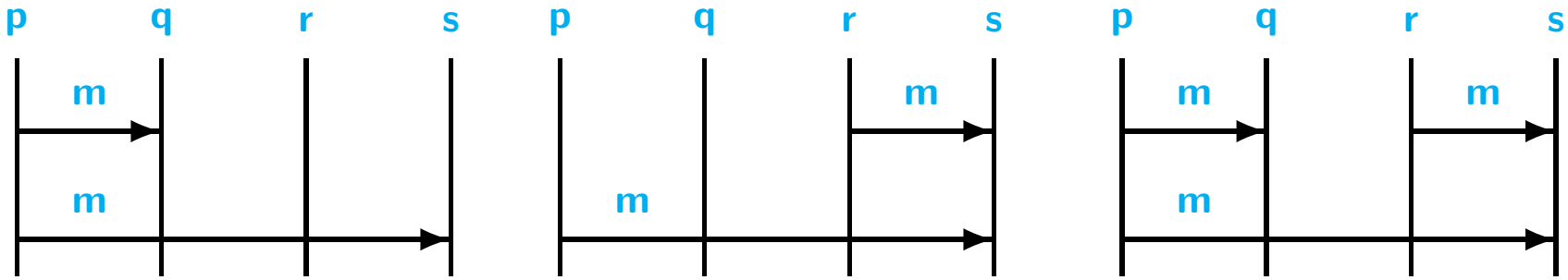
Implied scenarios [AEY00]



M_1 M_2 M

- **p** and **q** believe **M** is M_1
- **r** and **s** believe **M** is M_2

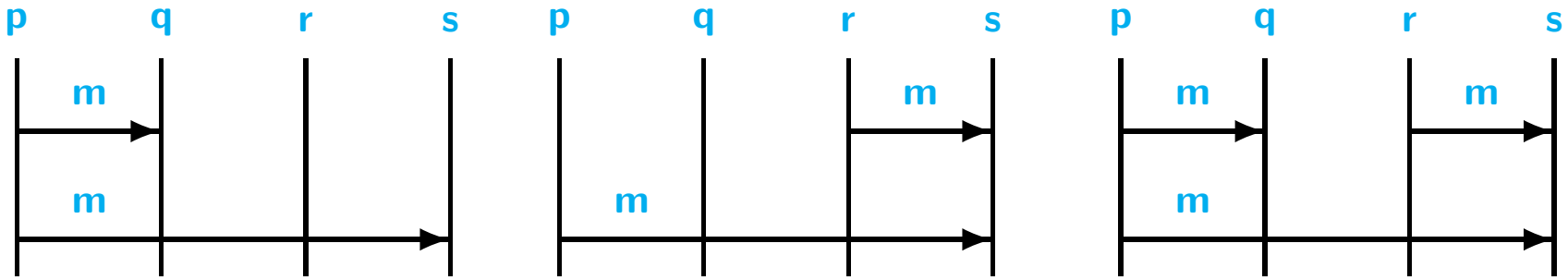
Implied scenarios [AEY00]



M_1 M_2 M

- **p** and **q** believe **M** is M_1
- **r** and **s** believe **M** is M_2
- MSC **M** is implied by **L** if for each process **p**, the **p**-projection of **M** matches the **p**-projection of some MSC in **L**

Implied scenarios [AEY00]



M_1 M_2 M

- **p** and **q** believe **M** is M_1
- **r** and **s** believe **M** is M_2
- MSC **M** is implied by **L** if for each process **p**, the **p**-projection of **M** matches the **p**-projection of some MSC in **L**
- An MSC language is locally testable if it is closed with respect to implied MSCs

Implied scenarios . . .

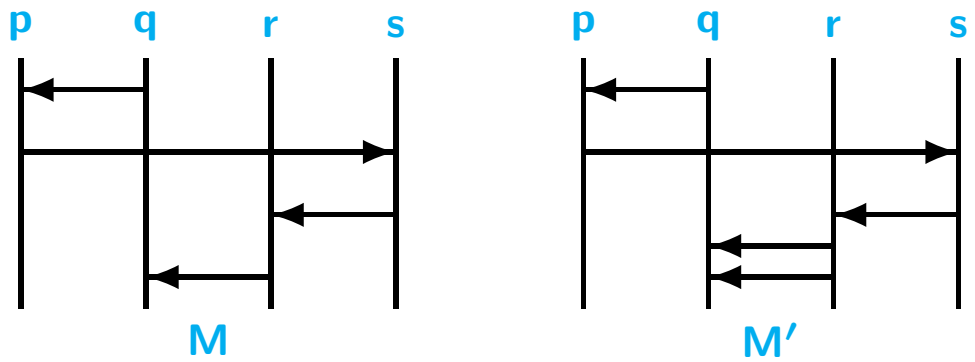
- Even for regular MSC languages, checking local testability is undecidable! [AEY01]

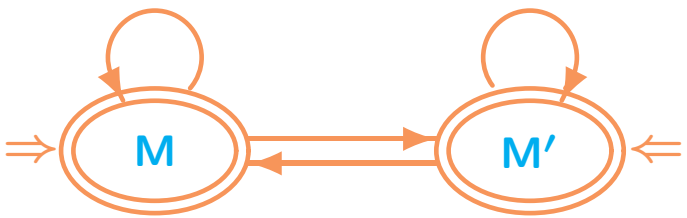
Implied scenarios . . .

- Even for regular MSC languages, checking local testability is undecidable! [AEY01]
- Even if the original language has bounded channels, its implied scenarios may not

Implied scenarios . . .

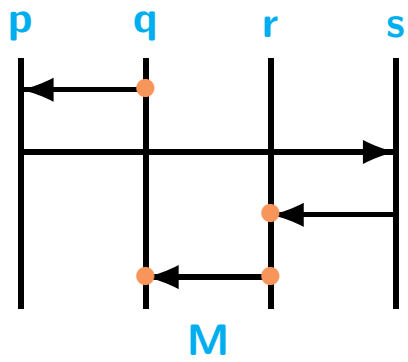
- Even for regular MSC languages, checking local testability is undecidable! [AEY01]
- Even if the original language has bounded channels, its implied scenarios may not

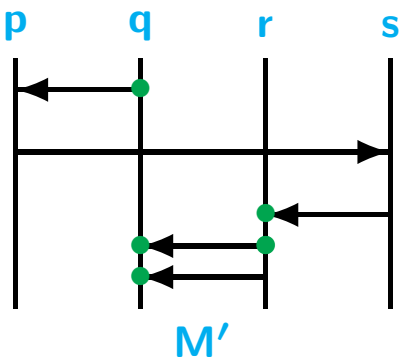




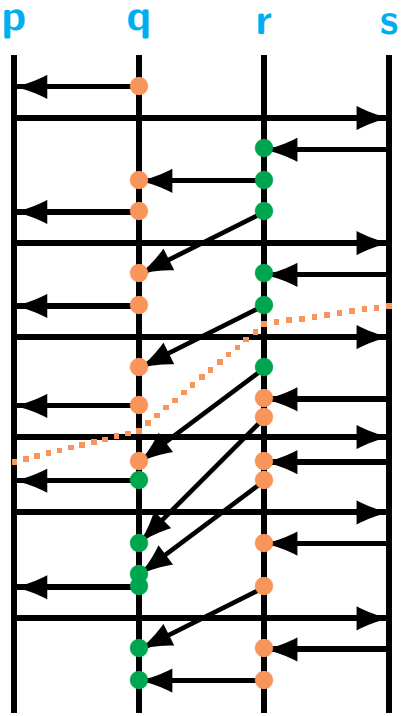
Implied scenarios . . .

- Even for regular MSC languages, checking local testability is undecidable! [AEY01]
- Even if the original language has bounded channels, its implied scenarios may not





Confusing $M^{2k}M'^{k}$ and $M'^{k}M^{2k}$ generates upto k messages in $p \longrightarrow s$ channel

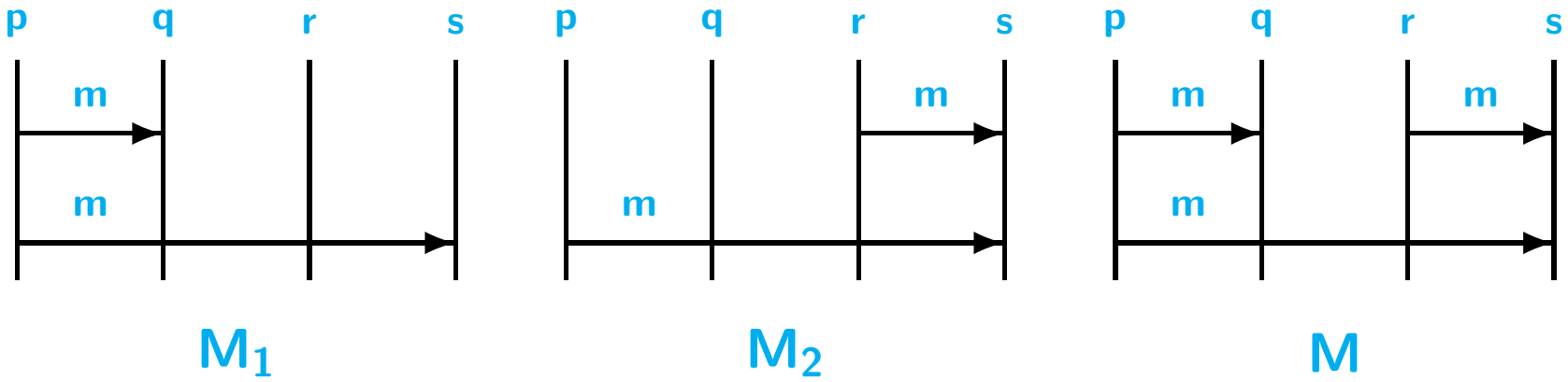


Joint observations

- What if we have observers who can record the behaviours of sets of processes?

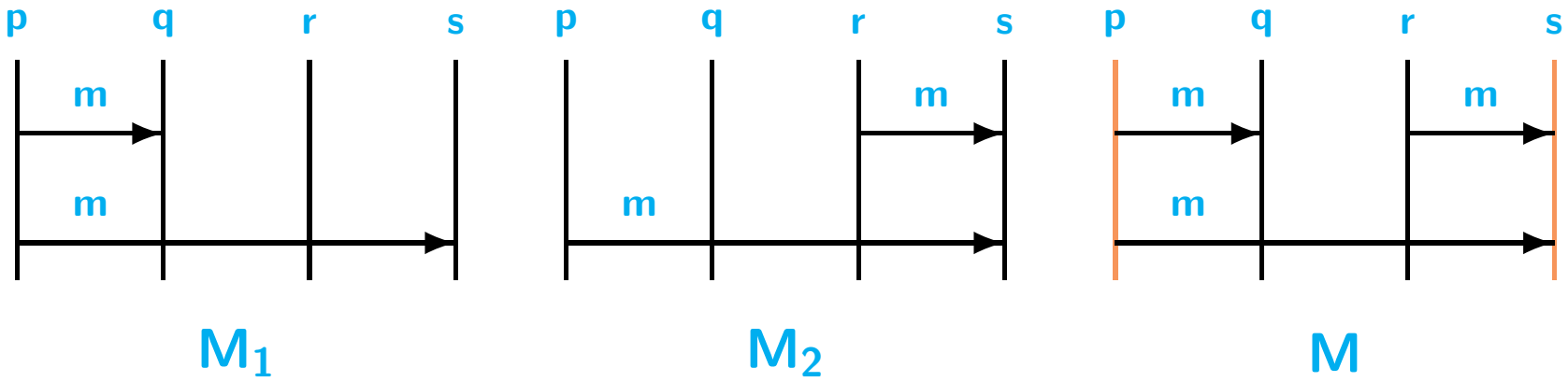
Joint observations

- What if we have observers who can record the behaviours of sets of processes?



Joint observations

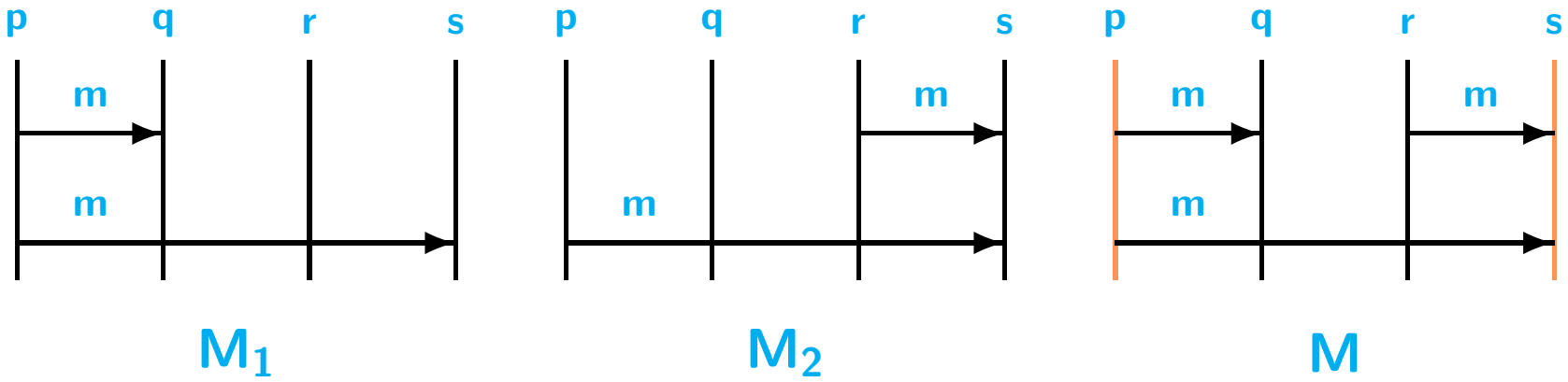
- What if we have observers who can record the behaviours of sets of processes?



- M is detected as an illegal MSC.
- Joint observers have more discriminating power.

Joint observations

- What if we have observers who can record the behaviours of sets of processes?



- M is detected as an illegal MSC.
- Joint observers have more discriminating power.
- The testing model of concurrent TTCN-3 corresponds to local testing with joint observers.

k-testability

- Record joint observations for every set P of processes of size k.

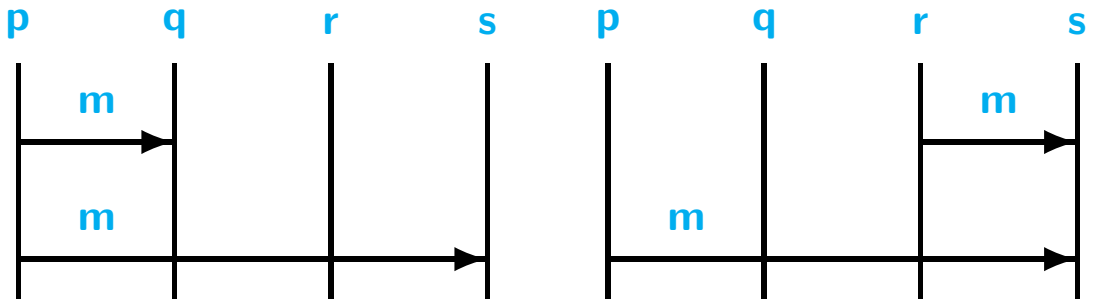
k-testability

- Record joint observations for every set **P** of processes of size **k**.
- **k**-closure of a language **L** $\triangleq$
$$\{\mathbf{M} \mid \forall \mathbf{P} \text{ s.t. } |\mathbf{P}| = \mathbf{k}.\ \exists \mathbf{M'} \in \mathbf{L}.\ \mathbf{M}{\restriction_{\mathbf{P}}} = \mathbf{M'}{\restriction_{\mathbf{P}}}\}$$
- **k**-implied scenario **M** for a language **L** $\triangleq$
MSC in the **k**-closure of **L** but not in **L**

k-testability

- Record joint observations for every set P of processes of size k.
- k-closure of a language $L \triangleq$
$$\{M \mid \forall P \text{ s.t. } |P| = k.\ \exists M' \in L.\ M{\restriction_P} = M'{\restriction_P}\}$$
- k-implied scenario M for a language $L \triangleq$
MSC in the k-closure of L but not in L
- A language is k-testable if it equals its k-closure
  - Local testability is 1-testability

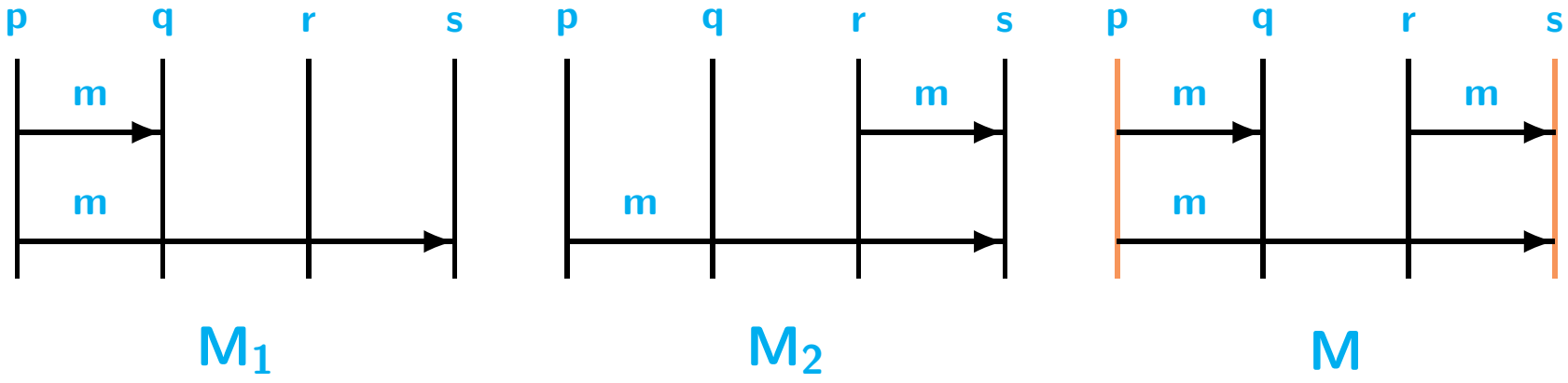
k-testability . . .



M_1 M_2

The set $\{M_1, M_2\}$ is 2-testable but not 1-testable.

k-testability . . .



M_1 M_2 M

The set $\{M_1, M_2\}$ is 2-testable but not 1-testable.

k-testability ...

- For all n and $k < n$ there are regular HMSC languages over n processes that are not k-testable
- 1-testability is undecidable for 4 or more processes. [AEY 01]
- n-testability is trivially decidable
  - What about k-testability for $1 < k < n$?
  - What is the smallest $k \leq n$ such that k-testability is decidable?

k-testability ...

- For all n and $k < n$ there are regular HMSC languages over n processes that are not k-testable
- 1-testability is undecidable for 4 or more processes. [AEY 01]
- n-testability is trivially decidable
  - What about k-testability for $1 < k < n$?
  - What is the smallest $k \leq n$ such that k-testability is decidable?

Theorem [BGMN, FCT 2007]

- k-testability is undecidable for $n \geq 3$ processes and $1 < k < n$
- 1-testability is undecidable for 2 processes
- k-testability remains undecidable even with a one-letter alphabet for $n \geq 3$ processes and $1 < k < n$

Local testing with tagging

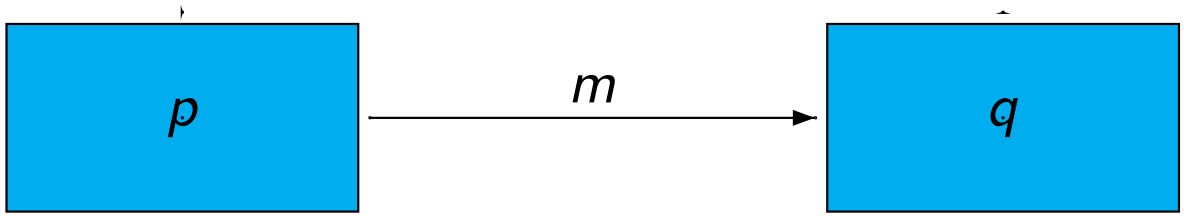
- Local testing of the original, unmodified system seems impossible

Local testing with tagging

- Local testing of the original, unmodified system seems impossible
- Embed system within testing layer that tags messages

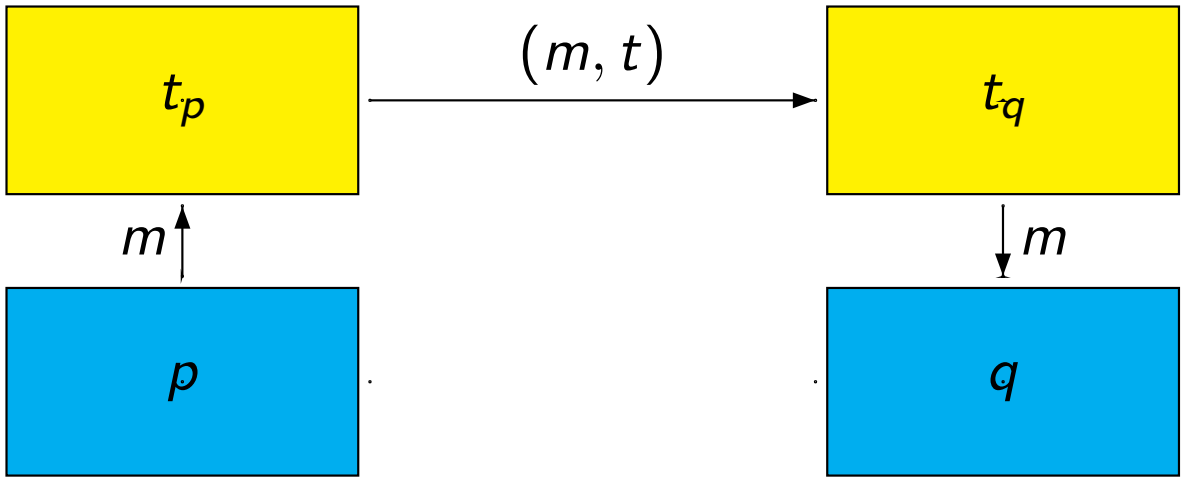
Local testing with tagging

- Local testing of the original, unmodified system seems impossible
- Embed system within testing layer that tags messages



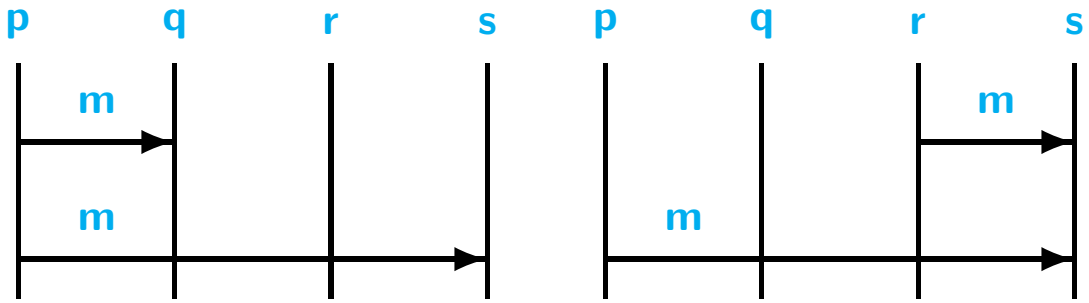
Local testing with tagging

- Local testing of the original, unmodified system seems impossible
- Embed system within testing layer that tags messages



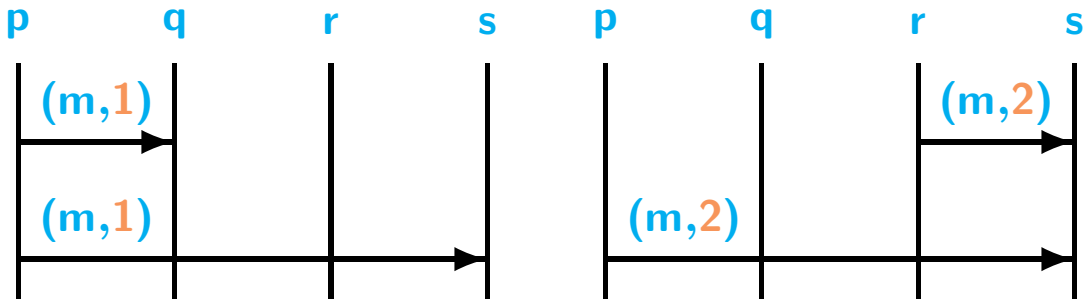
- Compare local observations of tagged system with tagged specification

Tagging for local testability . . .



M_1 M_2

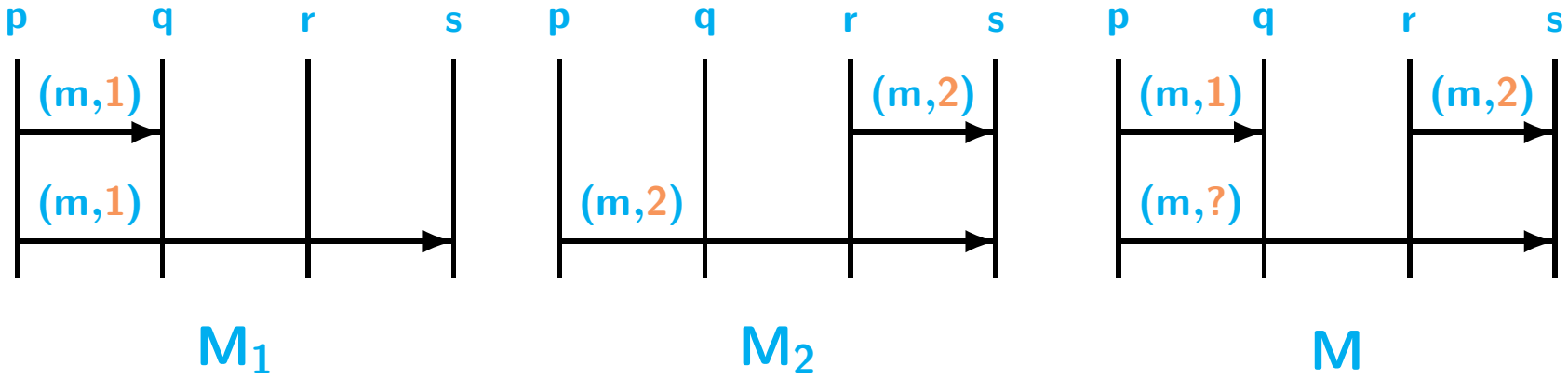
Tagging for local testability . . .



M_1 M_2

- By tagging auxiliary information to **m** in the specification, we know whether the message from **p** to **s** comes from M_1 or M_2

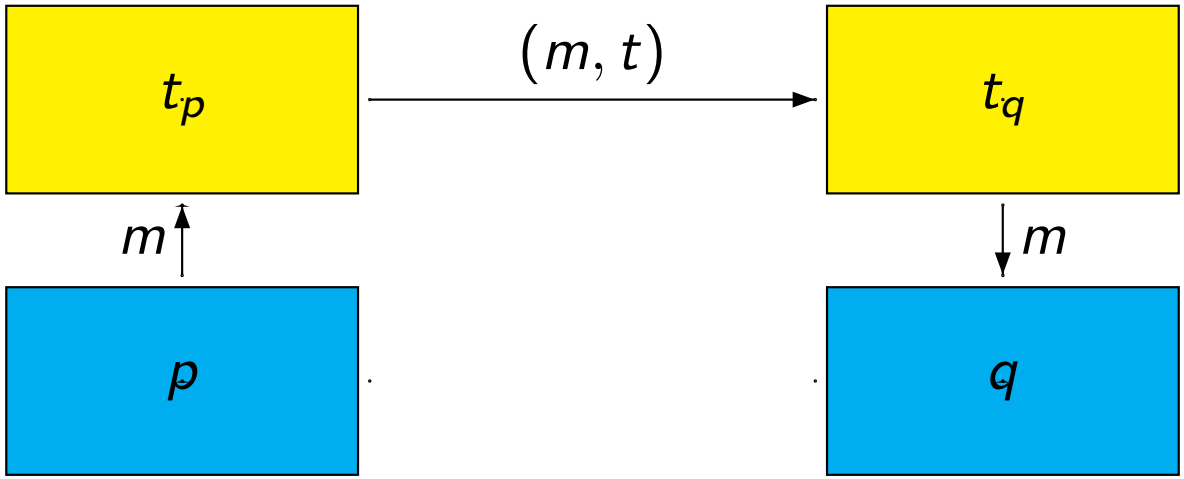
Tagging for local testability . . .



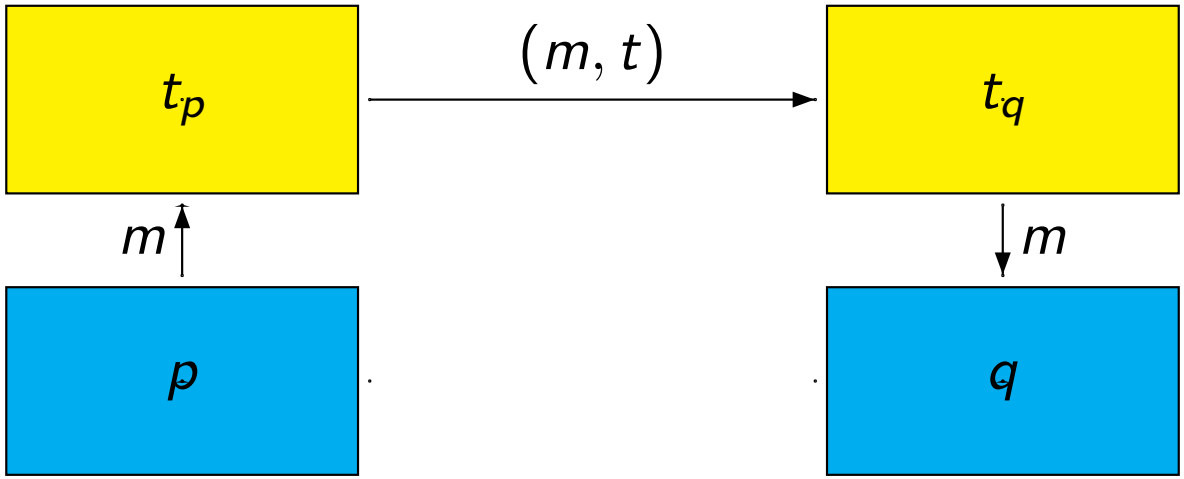
M_1 M_2 M

- By tagging auxiliary information to **m** in the specification, we know whether the message from **p** to **s** comes from M_1 or M_2
- This rules out the implied scenario **M**

Implementing tagging

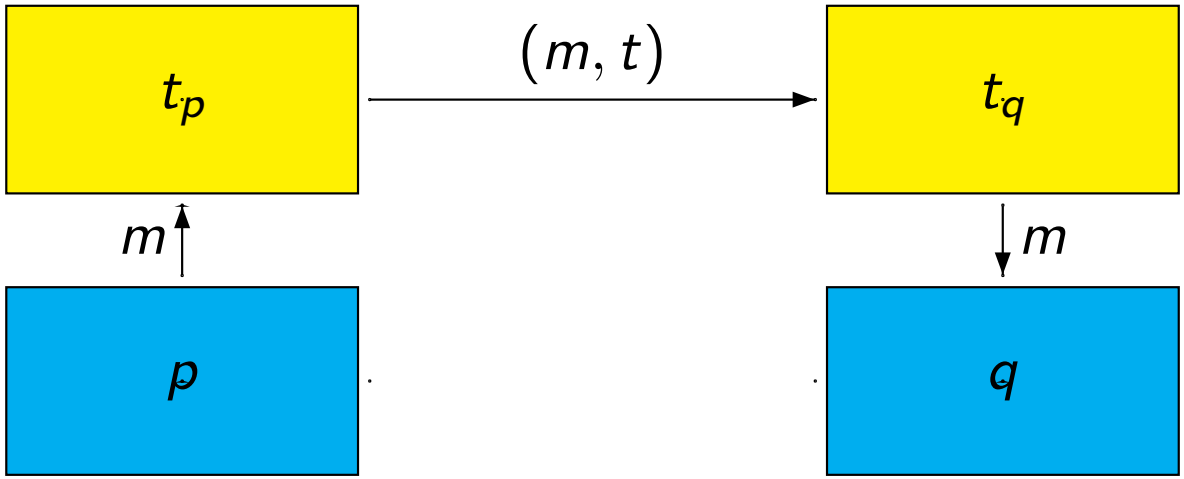


Implementing tagging



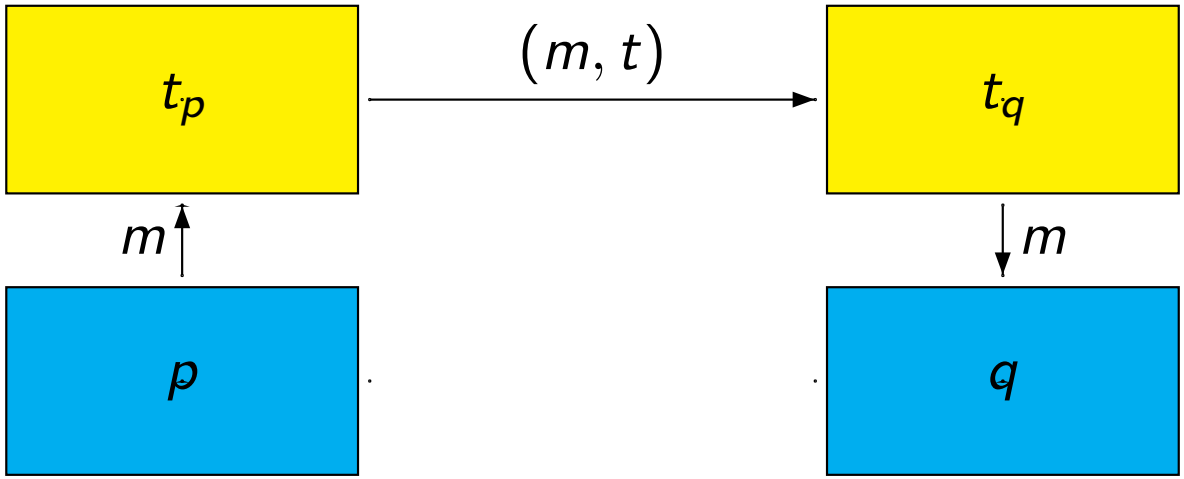
- Tagging is like adding a transport layer in a network

Implementing tagging



- Tagging is like adding a transport layer in a network
  - System under test is not modified
  - Underlying behaviour oblivious to tagging

Implementing tagging



- Tagging is like adding a transport layer in a network
  - System under test is not modified
  - Underlying behaviour oblivious to tagging
- Analogous to compiling with debug flag
  - Adding debugging information should not modify behaviour of program

Generating tags for local testing

- How do we generate tags for messages?

Generating tags for local testing

- How do we generate tags for messages?
- Bounded timestamping protocol [MNS03]
  - Each message is assigned a label (timestamp) from a bounded set
  - Augment messages with auxiliary information
    - Implicitly records information about size of channels
    - Includes timestamps of all unacknowledged messages
    - Message sent by **p** whose receive event is not in the causal past of **p**
  - Ensure that timestamps are reused in a consistent manner
  - If underlying computation has universally bounded channels, auxiliary information on each message is bounded

Local testing with timestamps

- Add timestamps to HMSC $\mathbf{H}$ to get "unfolded" HMSC $\mathbf{H}^{\mathsf{T}}$
  - Each node in $\mathbf{H}^{\mathsf{T}}$ is a timestamped copy of a node in $\mathbf{H}$
  - Edges in $\mathbf{H}^{\mathsf{T}}$ are consistent with $\mathbf{H}$ as well as timestamps
  - If $\mathbf{H}$ is locally synchronized, $\mathbf{H}^{\mathsf{T}}$ is finite

Local testing with timestamps

- Add timestamps to HMSC $\mathbf{H}$ to get "unfolded" HMSC $\mathbf{H}^{\mathsf{T}}$
  - Each node in $\mathbf{H}^{\mathsf{T}}$ is a timestamped copy of a node in $\mathbf{H}$
  - Edges in $\mathbf{H}^{\mathsf{T}}$ are consistent with $\mathbf{H}$ as well as timestamps
  - If $\mathbf{H}$ is locally synchronized, $\mathbf{H}^{\mathsf{T}}$ is finite
- System under test (SUT) is a communicating finite-state machine
  - Run SUT synchronously with bounded timestamping automaton

$$(q_s, q_t) \xrightarrow{(m,t)} (q_s', q_t'), \text{ provided}$$

$$q_s \xrightarrow{m} q_s' \text{ (SUT transition), and}$$

$$q_t \xrightarrow{t} q_t' \text{ (Timestamp transition)}$$

Local testing with timestamps

- Add timestamps to HMSC $\mathbf{H}$ to get "unfolded" HMSC $\mathbf{H}^{\mathbf{T}}$
  - Each node in $\mathbf{H}^{\mathbf{T}}$ is a timestamped copy of a node in $\mathbf{H}$
  - Edges in $\mathbf{H}^{\mathbf{T}}$ are consistent with $\mathbf{H}$ as well as timestamps
  - If $\mathbf{H}$ is locally synchronized, $\mathbf{H}^{\mathbf{T}}$ is finite
- System under test (SUT) is a communicating finite-state machine
  - Run SUT synchronously with bounded timestamping automaton

$$(q_s, q_t) \xrightarrow{(m,t)} (q'_s, q'_t), \text{ provided}$$

$$q_s \xrightarrow{m} q'_s \text{ (SUT transition), and}$$

$$q_t \xrightarrow{t} q'_t \text{ (Timestamp transition)}$$

- Compare timestamped runs of SUT with $\mathbf{H}^{\mathbf{T}}$

$\mathbf{H}^{\mathsf{T}}$ and implied scenarios

Theorem

If $\mathbf{H}$ is locally synchronized, implied scenarios for $\mathbf{H}^{\mathsf{T}}$ have universally bounded channels

$\mathbf{H}^{\mathsf{T}}$ and implied scenarios

Theorem

If $\mathbf{H}$ is locally synchronized, implied scenarios for $\mathbf{H}^{\mathsf{T}}$ have universally bounded channels

- Implied scenarios have internally consistent timestamps
  - Timestamping protocol is deterministic
  - Re-timestamping an implied scenario generates identical timestamps

$\mathbf{H}^{\mathbf{T}}$ and implied scenarios

Theorem

If $\mathbf{H}$ is locally synchronized, implied scenarios for $\mathbf{H}^{\mathbf{T}}$ have universally bounded channels

- Implied scenarios have internally consistent timestamps
  - Timestamping protocol is deterministic
  - Re-timestamping an implied scenario generates identical timestamps
- If some channel in an implied scenario exceeds the universal bound $\mathbf{B}$ of $\mathbf{H}^{\mathbf{T}}$, the timestamp will have more than $\mathbf{B}$ unacknowledged messages for that channel

$\mathbf{H}^{\mathbf{T}}$ and implied scenarios

Theorem

If $\mathbf{H}$ is locally synchronized, implied scenarios for $\mathbf{H}^{\mathbf{T}}$ have universally bounded channels

- Implied scenarios have internally consistent timestamps
  - Timestamping protocol is deterministic
  - Re-timestamping an implied scenario generates identical timestamps
- If some channel in an implied scenario exceeds the universal bound $\mathbf{B}$ of $\mathbf{H}^{\mathbf{T}}$, the timestamp will have more than $\mathbf{B}$ unacknowledged messages for that channel
- This timestamp cannot be consistent with $\mathbf{H}^{\mathbf{T}}$

$\mathbf{H}^{\mathsf{T}}$ and local testability

Corollary

If $\mathbf{H}$ is locally synchronized, then checking whether $\mathbf{H}^{\mathsf{T}}$ is locally testable is decidable

$\mathbf{H}^{\mathbf{T}}$ and local testability

Corollary

If $\mathbf{H}$ is locally synchronized, then checking whether $\mathbf{H}^{\mathbf{T}}$ is locally testable is decidable

- Project $\mathbf{H}^{\mathbf{T}}$ onto each process $\mathbf{p}$
  - Projections describe a CFM $\mathbf{A}^{\mathbf{T}}$ for implied scenarios of $\mathbf{H}^{\mathbf{T}}$
  - This CFM has universally bounded channels
  - Implied scenarios of $\mathbf{H}^{\mathbf{T}}$ form a regular MSC language

$\mathbf{H}^{\mathbf{T}}$ and local testability

Corollary

If $\mathbf{H}$ is locally synchronized, then checking whether $\mathbf{H}^{\mathbf{T}}$ is locally testable is decidable

- Project $\mathbf{H}^{\mathbf{T}}$ onto each process $\mathbf{p}$
  - Projections describe a CFM $\mathbf{A}^{\mathbf{T}}$ for implied scenarios of $\mathbf{H}^{\mathbf{T}}$
  - This CFM has universally bounded channels
  - Implied scenarios of $\mathbf{H}^{\mathbf{T}}$ form a regular MSC language
- Construct a CFM $\mathbf{A}$ from $\mathbf{H}^{\mathbf{T}}$

$\mathbf{H}^{\mathbf{T}}$ and local testability

Corollary

If $\mathbf{H}$ is locally synchronized, then checking whether $\mathbf{H}^{\mathbf{T}}$ is locally testable is decidable

- Project $\mathbf{H}^{\mathbf{T}}$ onto each process $\mathbf{p}$
  - Projections describe a CFM $\mathbf{A}^{\mathbf{T}}$ for implied scenarios of $\mathbf{H}^{\mathbf{T}}$
  - This CFM has universally bounded channels
  - Implied scenarios of $\mathbf{H}^{\mathbf{T}}$ form a regular MSC language
- Construct a CFM $\mathbf{A}$ from $\mathbf{H}^{\mathbf{T}}$
- Compare $\mathbf{L(A)}$ and $\mathbf{L(A^{T})}$
  - Both define regular MSC languages

Connections to realizability

- Implied scenarios arise in the context of realiability [AEY00]
  - Naively synthesize CFM as product of MSC projections

Connections to realizability

- Implied scenarios arise in the context of realiability [AEY00]
  - Naively synthesize CFM as product of MSC projections
- Synchronous setting : Can cover MSC specification with minimum implied scenarios [UKM01]

Connections to realizability

- Implied scenarios arise in the context of realiability [AEY00]
  - Naively synthesize CFM as product of MSC projections
- Synchronous setting : Can cover MSC specification with minimum implied scenarios [UKM01]
- Asynchronous setting : Can cover arbitrary regular MSC specifications exactly using complex timestamping [HMNST05]

Connections to realizability

- Implied scenarios arise in the context of realiability [AEY00]
  - Naively synthesize CFM as product of MSC projections
- Synchronous setting : Can cover MSC specification with minimum implied scenarios [UKM01]
- Asynchronous setting : Can cover arbitrary regular MSC specifications exactly using complex timestamping [HMNST05]
- Tagging can offer an intermediate solution
  - Synthesize CFM by combining projections modulo tagging
  - Tagging is deadlock free — will produce a deadlock free implementation if one exists

Connection with causal closure

- Timestamping approach motivated by work on causal closure of HMSCs [AMNN05]
- Causal view of a process **p** of MSC **M** is set of events causally below maximum **p** event in **M**
- Causally implied MSC — causal view of each process is consistent with some MSC in the language
- Casual closure of a regular MSC language is regular
  - Use timestamping to show this, but . . .
  - . . . timestamps must be integrated into CFM
- For local testing, timestamping is oblivious to structure of CFM

Other tagging schemes

- Can we enhance timestamping to recover causal closure?

Other tagging schemes

- Can we enhance timestamping to recover causal closure?
- Should we allow timestamps to depend on HMSC?
  - "Transport layer" for tagging depends on specification
  - Non-uniform test procedure

Other tagging schemes

- Can we enhance timestamping to recover causal closure?
- Should we allow timestamps to depend on HMSC?
  - "Transport layer" for tagging depends on specification
  - Non-uniform test procedure
- Maximally discriminating tagging schemes vs minimally implementable ones
- Complex timestamps blow up the specification
  - Increases the lengths the local tests to be performed

Other tagging schemes

- Can we enhance timestamping to recover causal closure?
- Should we allow timestamps to depend on HMSC?
  - "Transport layer" for tagging depends on specification
  - Non-uniform test procedure
- Maximally discriminating tagging schemes vs minimally implementable ones
- Complex timestamps blow up the specification
  - Increases the lengths the local tests to be performed
- Can we design more efficient tagging schemes?
  - For instance, one tag per node in the HMSC
- Implementation under test is untagged
  - Need to be able to generate tags on the fly while testing

Summary and future work

- HMSCs — mechanism for specifying collections of scenarios
  - Attractive visual formalism
  - Sufficient condition for regularity
- Local testability is generally undecidable for the original specification
- Tagging specification with timestamps make local testability decidable
- What is the "optimal" way to tag a system?
  - Adding expressiveness increases complexity of testing
- Can we design more efficient tagging schemes?
  - Need to generate tags on the fly