The Theory of Message Sequence Charts – II

K Narayan Kumar

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

TIFR, Mumbai, 01 May, 2009

Summary

- MSCs describe runs or behaviours of message passing systems.

Summary

- MSCs describe runs or behaviours of message passing systems.
- MSGs, a visual formalism to describe languages of MSCs.

Summary

- MSCs describe runs or behaviours of message passing systems.
- MSGs, a visual formalism to describe languages of MSCs.
- MSGs : regularity is not decidable, but boundedness is.

Summary

- MSCs describe runs or behaviours of message passing systems.
- MSGs, a visual formalism to describe languages of MSCs.
- MSGs : regularity is not decidable, but boundedness is.
- MPAs : An operational model, distributed, ...

Summary

- MSCs describe runs or behaviours of message passing systems.
- MSGs, a visual formalism to describe languages of MSCs.
- MSGs : regularity is not decidable, but boundedness is.
- MPAs : An operational model, distributed, ...
- Verifying implementability for MSGs is undecidable.

The Model-checking problem

- Positive Model-checking Given a specification language S and an implementation L decide if $L \subseteq S$.

The Model-checking problem

- Positive Model-checking Given a specification language S and an implementation L decide if $L \subseteq S$.

  Are all the positive instances exhibited?

The Model-checking problem

- Positive Model-checking Given a specification language S and an implementation L decide if $L \subseteq S$.

  Are all the positive instances exhibited?

- Negative Model-checking Given a specification language S and an implementation L decide whether $S \cap L = \emptyset$.

The Model-checking problem

- Positive Model-checking Given a specification language S and an implementation L decide if $L \subseteq S$.

  Are all the positive instances exhibited?

- Negative Model-checking Given a specification language S and an implementation L decide whether $S \cap L = \emptyset$.

  Are all the negative instances avoided?

The Model-checking problem ...

The Model-checking problem ...

- If S and L are given as locally synchronized MSGs, both the model checking problems are decidable.

The Model-checking problem ...

- If S and L are given as locally synchronized MSGs, both the model checking problems are decidable.
- If S is given by a locally synchronized MSG and L is given by any MSG, both the model checking problems are decidable.

The Model-checking problem ...

- If S and L are given as locally synchronized MSGs, both the model checking problems are decidable.
- If S is given by a locally synchronized MSG and L is given by any MSG, both the model checking problems are decidable.
    1. Replace each node in the MSG with a linearization.

The Model-checking problem ...

- If S and L are given as locally synchronized MSGs, both the model checking problems are decidable.
- If S is given by a locally synchronized MSG and L is given by any MSG, both the model checking problems are decidable.
  1. Replace each node in the MSG with a linearization.
  2. Let X be the regular language accepted by the resulting finite automaton.

The Model-checking problem ...

- If S and L are given as locally synchronized MSGs, both the model checking problems are decidable.
- If S is given by a locally synchronized MSG and L is given by any MSG, both the model checking problems are decidable.
  1. Replace each node in the MSG with a linearization.
  2. Let X be the regular language accepted by the resulting finite automaton.
  3. $L \subseteq S$ if and only if $X \subseteq S$ and $L \cap S = \emptyset$ if and only if $X \cap S = \emptyset$.

The Model-checking problem ...

- If S and L are given as locally synchronized MSGs, both the model checking problems are decidable.
- If S is given by a locally synchronized MSG and L is given by any MSG, both the model checking problems are decidable.
  1. Replace each node in the MSG with a linearization.
  2. Let X be the regular language accepted by the resulting finite automaton.
  3. $L \subseteq S$ if and only if $X \subseteq S$ and $L \cap S = \emptyset$ if and only if $X \cap S = \emptyset$.
- These results can be generalized further ...

Model-checking ...

Sufficient conditions for the decidability of model-checking:

Model-checking ...

Sufficient conditions for the decidability of model-checking:

- The system L has a regular set of representatives.

  A regular language R such that the set of MSCs generated by the words in R is L.

Model-checking ...

Sufficient conditions for the decidability of model-checking:

- The system L has a regular set of representatives.

  A regular language R such that the set of MSCs generated by the words in R is L.

- Given B, we can effectively construct $Lin^B(S)$ consisiting of all the B bounded linearizations of MSCs in S.

Globally Cooperative MSGs

Globally Cooperative MSGs

- An MSC is globally cooperative, if the symmetric closure of its communication graph has a single nontrivial SCC.

Globally Cooperative MSGs

- An MSC is globally cooperative, if the symmetric closure of its communication graph has a single nontrivial SCC.
- An MSG is globally cooperative, if every loop in the MSG generates a globally cooperative MSC.

Globally Cooperative MSGs

- An MSC is globally cooperative, if the symmetric closure of its communication graph has a single nontrivial SCC.
- An MSG is globally cooperative, if every loop in the MSG generates a globally cooperative MSC.

Rules out *independent iterations* without insisting on regularity.

Globally Cooperative MSGs

- An MSC is globally cooperative, if the symmetric closure of its communication graph has a single nontrivial SCC.
- An MSG is globally cooperative, if every loop in the MSG generates a globally cooperative MSC.

Rules out *independent iterations* without insisting on regularity.

Theorem: Given B, the set of B bounded linearizations of a GC-MSG is a regular language.

Globally Cooperative MSGs

- An MSC is globally cooperative, if the symmetric closure of its communication graph has a single nontrivial SCC.
- An MSG is globally cooperative, if every loop in the MSG generates a globally cooperative MSC.

Rules out *independent iterations* without insisting on regularity.

Theorem: Given B, the set of B bounded linearizations of a GC-MSG is a regular language.

[One of the many results best proved via a translation to Mazurkiewicz traces.]

Globally Cooperative MSGs

- An MSC is globally cooperative, if the symmetric closure of its communication graph has a single nontrivial SCC.
- An MSG is globally cooperative, if every loop in the MSG generates a globally cooperative MSC.

Rules out *independent iterations* without insisting on regularity.

Theorem: Given B, the set of B bounded linearizations of a GC-MSG is a regular language.

[One of the many results best proved via a translation to Mazurkiewicz traces.]

Corollary: Systems given as MSGs can be model-checked w.r.t. specifications presented as GC-MSGs.

Implementing Regular MSC languages

- Allow global accepting states.

Implementing Regular MSC languages

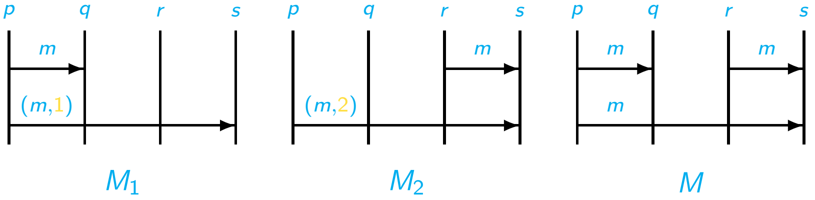
- Allow global accepting states.
- Allow tagging of messages with additional content.

Implementing Regular MSC languages

- Allow global accepting states.
- Allow tagging of messages with additional content.
- No additional messages.

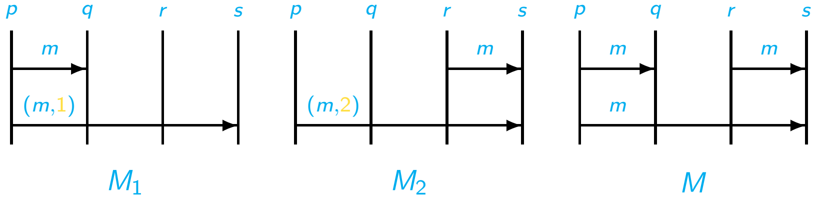
Implementing Regular MSC languages

- Allow global accepting states.
- Allow tagging of messages with additional content.
- No additional messages.



Implementing Regular MSC languages

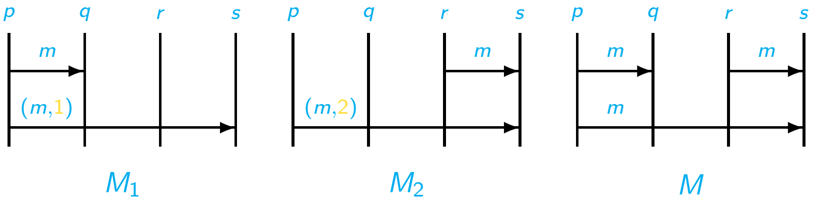
- Allow global accepting states.
- Allow tagging of messages with additional content.
- No additional messages.



- By tagging auxiliary information to m, p informs s whether it has sent a message to q

Implementing Regular MSC languages

- Allow global accepting states.
- Allow tagging of messages with additional content.
- No additional messages.



- By tagging auxiliary information to m, p informs s whether it has sent a message to q
- This rules out the implied scenario M

Distributed Synthesis

Distributed Synthesis

Theorem: An MSC language L is regular if and only if there is an an MPA A, with a finite auxiliary message alphabet Δ, that accepts L.

Distributed Synthesis

Theorem: An MSC language L is regular if and only if there is an an MPA A, with a finite auxiliary message alphabet Δ, that accepts L.

- As a matter of fact one can construct a deterministic MPA accepting A.

Distributed Synthesis

Theorem: An MSC language L is regular if and only if there is an an MPA A, with a finite auxiliary message alphabet Δ, that accepts L.

- As a matter of fact one can construct a deterministic MPA accepting A.
- One proof is to translate MSC languages to trace languages, use Zielonka's theorem and then describe a deterministic simulation of the resulting Asynchronous automaton using MPAs.

Distributed Synthesis

Theorem: An MSC language L is regular if and only if there is an an MPA A, with a finite auxiliary message alphabet Δ, that accepts L.

- As a matter of fact one can construct a deterministic MPA accepting A.
- One proof is to translate MSC languages to trace languages, use Zielonka's theorem and then describe a deterministic simulation of the resulting Asynchronous automaton using MPAs.
- Explicitely construct a deterministic MPA from a FA accepting the linearizations of L.

Synthesis ...

- Let A be finite automaton accepting the linearizations of L.

Synthesis ...

- Let A be finite automaton accepting the linearizations of L.
- How does the MPA maintain the state of this automaton A as it reads an MSC?

Synthesis ...

- Let A be finite automaton accepting the linearizations of L.
- How does the MPA maintain the state of this automaton A as it reads an MSC?
- No process sees the entire past. Each process has information only on part of the MSC.

Synthesis ...

- Let A be finite automaton accepting the linearizations of L.
- How does the MPA maintain the state of this automaton A as it reads an MSC?
- No process sees the entire past. Each process has information only on part of the MSC.
- Instead of maintining a word or an MSC keep the function on the states of A defined by this word (or MSC).

Synthesis ...

- Let A be finite automaton accepting the linearizations of L.
- How does the MPA maintain the state of this automaton A as it reads an MSC?
- No process sees the entire past. Each process has information only on part of the MSC.
- Instead of maintining a word or an MSC keep the function on the states of A defined by this word (or MSC).
- Putting together partial information

Synthesis ...

- Let A be finite automaton accepting the linearizations of L.
- How does the MPA maintain the state of this automaton A as it reads an MSC?
- No process sees the entire past. Each process has information only on part of the MSC.
- Instead of maintining a word or an MSC keep the function on the states of A defined by this word (or MSC).
- Putting together partial information
  1. Process 1 keeps the effect of the MSC in its past.

Synthesis ...

- Let A be finite automaton accepting the linearizations of L.
- How does the MPA maintain the state of this automaton A as it reads an MSC?
- No process sees the entire past. Each process has information only on part of the MSC.
- Instead of maintining a word or an MSC keep the function on the states of A defined by this word (or MSC).
- Putting together partial information
  1. Process 1 keeps the effect of the MSC in its past.
  2. Process 2 keeps the effect of the partial MSC consisting those events seen by 2 but not 1.

Synthesis ...

- Let A be finite automaton accepting the linearizations of L.
- How does the MPA maintain the state of this automaton A as it reads an MSC?
- No process sees the entire past. Each process has information only on part of the MSC.
- Instead of maintining a word or an MSC keep the function on the states of A defined by this word (or MSC).
- Putting together partial information
  1. Process 1 keeps the effect of the MSC in its past.
  2. Process 2 keeps the effect of the partial MSC consisting those events seen by 2 but not 1.
  3. Process 3 keeps the effect of the partial MSC consisting of events seen by 3 but not by 2 and 1.

Synthesis ...

- Let A be finite automaton accepting the linearizations of L.
- How does the MPA maintain the state of this automaton A as it reads an MSC?
- No process sees the entire past. Each process has information only on part of the MSC.
- Instead of maintining a word or an MSC keep the function on the states of A defined by this word (or MSC).
- Putting together partial information
  1. Process 1 keeps the effect of the MSC in its past.
  2. Process 2 keeps the effect of the partial MSC consisting those events seen by 2 but not 1.
  3. Process 3 keeps the effect of the partial MSC consisting of events seen by 3 but not by 2 and 1.
  4. ...

Synthesis ...

- Let A be finite automaton accepting the linearizations of L.
- How does the MPA maintain the state of this automaton A as it reads an MSC?
- No process sees the entire past. Each process has information only on part of the MSC.
- Instead of maintining a word or an MSC keep the function on the states of A defined by this word (or MSC).
- Putting together partial information
  1. Process 1 keeps the effect of the MSC in its past.
  2. Process 2 keeps the effect of the partial MSC consisting those events seen by 2 but not 1.
  3. Process 3 keeps the effect of the partial MSC consisting of events seen by 3 but not by 2 and 1.
  4. ...
- A sophisticated local timestamping algorithm is needed to make all this work.

Monadic Second Order Logic

The Monadic Second Order logic over MSCs.

- $x, y, \ldots$ an infinite collection of first-order variables.

Monadic Second Order Logic

The Monadic Second Order logic over MSCs.

- $x, y, \ldots$ an infinite collection of first-order variables.
- $X, Y, \ldots$ an infinite collection of second-order set variables.

Monadic Second Order Logic

The Monadic Second Order logic over MSCs.

- $x, y, \ldots$ an infinite collection of first-order variables.
- $X, Y, \ldots$ an infinite collection of second-order set variables.
- Atomic Formulas

Monadic Second Order Logic

The Monadic Second Order logic over MSCs.

- $x, y, \ldots$ an infinite collection of first-order variables.
- $X, Y, \ldots$ an infinite collection of second-order set variables.
- Atomic Formulas
  - $\lambda(x)$ where λ is an action

Monadic Second Order Logic

The Monadic Second Order logic over MSCs.

- $x, y, \dots$ an infinite collection of first-order variables.
- $X, Y, \dots$ an infinite collection of second-order set variables.
- Atomic Formulas
  - $\lambda(x)$ where λ is an action
  - $x \in X$

Monadic Second Order Logic

The Monadic Second Order logic over MSCs.

- $x, y, \ldots$ an infinite collection of first-order variables.
- $X, Y, \ldots$ an infinite collection of second-order set variables.
- Atomic Formulas
  - $\lambda(x)$ where λ is an action
  - $x \in X$
  - $x \leq y$

Monadic Second Order Logic

The Monadic Second Order logic over MSCs.

- $x, y, \ldots$ an infinite collection of first-order variables.
- $X, Y, \ldots$ an infinite collection of second-order set variables.
- Atomic Formulas
  - $\lambda(x)$ where λ is an action
  - $x \in X$
  - $x \leq y$
  - $x <_m y$

Monadic Second Order Logic

The Monadic Second Order logic over MSCs.

- $x, y, \ldots$ an infinite collection of first-order variables.
- $X, Y, \ldots$ an infinite collection of second-order set variables.
- Atomic Formulas
  - $\lambda(x)$ where λ is an action
  - $x \in X$
  - $x \leq y$
  - $x <_m y$
- Quantification and boolean connectives.

Monadic Second Order Logic

The Monadic Second Order logic over MSCs.

- $x, y, \ldots$ an infinite collection of first-order variables.
- $X, Y, \ldots$ an infinite collection of second-order set variables.
- Atomic Formulas
  - $\lambda(x)$ where λ is an action
  - $x \in X$
  - $x \leq y$
  - $x <_m y$
- Quantification and boolean connectives.

Monadic Second Order Logic

The Monadic Second Order logic over MSCs.

- $x, y, \ldots$ an infinite collection of first-order variables.
- $X, Y, \ldots$ an infinite collection of second-order set variables.
- Atomic Formulas
  - $\lambda(x)$ where λ is an action
  - $x \in X$
  - $x \leq y$
  - $x <_m y$
- Quantification and boolean connectives.
- The first order variables take values over the events in the given MSC.

Monadic Second Order Logic

The Monadic Second Order logic over MSCs.

- $x, y, \ldots$ an infinite collection of first-order variables.
- $X, Y, \ldots$ an infinite collection of second-order set variables.
- Atomic Formulas
  - $\lambda(x)$ where λ is an action
  - $x \in X$
  - $x \leq y$
  - $x <_m y$
- Quantification and boolean connectives.
- The first order variables take values over the events in the given MSC.
- The second order variables take subsets of events as values.

Monadic Second Order Logic

The Monadic Second Order logic over MSCs.

- $x, y, \ldots$ an infinite collection of first-order variables.
- $X, Y, \ldots$ an infinite collection of second-order set variables.
- Atomic Formulas
  - $\lambda(x)$ where λ is an action
  - $x \in X$
  - $x \leq y$
  - $x <_m y$
- Quantification and boolean connectives.
- The first order variables take values over the events in the given MSC.
- The second order variables take subsets of events as values.
- $\leq$ is interpreted by the ordering on the MSC.

Monadic Second Order Logic

The Monadic Second Order logic over MSCs.

- $x, y, \ldots$ an infinite collection of first-order variables.
- $X, Y, \ldots$ an infinite collection of second-order set variables.
- Atomic Formulas
  - $\lambda(x)$ where λ is an action
  - $x \in X$
  - $x \leq y$
  - $x <_m y$
- Quantification and boolean connectives.
- The first order variables take values over the events in the given MSC.
- The second order variables take subsets of events as values.
- $\leq$ is interpreted by the ordering on the MSC.
- $<_m$ denotes the message ordering and cannot be defined using $\leq$

MSO ...

$$\forall \mathbf{x}.\, \forall \mathbf{y}.\ (\mathbf{p!q}(\mathbf{x}) \wedge \mathbf{p!q}(\mathbf{y}) \wedge (\mathbf{x} < \mathbf{y})) \implies \exists \mathbf{z}.(\mathbf{x} <_{\mathbf{m}} \mathbf{z}) \wedge (\mathbf{z} \leq \mathbf{y}).$$

MSO ...

$$\forall \mathbf{x}.\ \forall \mathbf{y}.\ (\mathbf{p!q(x)} \wedge \mathbf{p!q(y)} \wedge (\mathbf{x} < \mathbf{y})) \implies \exists \mathbf{z}.(\mathbf{x} <_{\mathbf{m}} \mathbf{z}) \wedge (\mathbf{z} \leq \mathbf{y}).$$

- p sends to q only after receiving an (indirect) acknowedgement for the previous send.

MSO ...

$$\forall \mathbf{x}.\ \forall \mathbf{y}.\ (\mathbf{p!q}(\mathbf{x}) \wedge \mathbf{p!q}(\mathbf{y}) \wedge (\mathbf{x} < \mathbf{y})) \implies \exists \mathbf{z}.(\mathbf{x} <_{\mathbf{m}} \mathbf{z}) \wedge (\mathbf{z} \leq \mathbf{y}).$$

- p sends to q only after receiving an (indirect) acknowledgement for the previous send.

The channel from p to q is universally 1*-bounded.*

MSO ...

$$\forall \mathbf{x}.\ \forall \mathbf{y}.\ (\mathbf{p!q}(\mathbf{x}) \wedge \mathbf{p!q}(\mathbf{y}) \wedge (\mathbf{x} < \mathbf{y})) \implies \exists \mathbf{z}.(\mathbf{x} <_{\mathbf{m}} \mathbf{z}) \wedge (\mathbf{z} \leq \mathbf{y}).$$

- p sends to q only after receiving an (indirect) acknowledgement for the previous send.

  The channel from p to q is universally 1*-bounded.*

- One can express B-boundedness for any B.

MSO ...

$$\forall \mathbf{x}.\ \forall \mathbf{y}.\ (\mathbf{p!q(x)} \wedge \mathbf{p!q(y)} \wedge (\mathbf{x} < \mathbf{y})) \implies \exists \mathbf{z}.(\mathbf{x} <_{\mathbf{m}} \mathbf{z}) \wedge (\mathbf{z} \leq \mathbf{y}).$$

- p sends to q only after receiving an (indirect) acknowledgement for the previous send.

  The channel from p to q is universally 1*-bounded.*

- One can express B-boundedness for any B.

MSO ...

$$\forall \mathbf{x}.\ \forall \mathbf{y}.\ (\mathbf{p!q(x)} \wedge \mathbf{p!q(y)} \wedge (\mathbf{x} < \mathbf{y})) \implies \exists \mathbf{z}.(\mathbf{x} <_{\mathbf{m}} \mathbf{z}) \wedge (\mathbf{z} \leq \mathbf{y}).$$

- p sends to q only after receiving an (indirect) acknowedgement for the previous send.

The channel from p to q is universally 1-bounded.

- One can express B-boundedness for any B.

Theorem: An MSC language L is regular if and only if there is a formula φ in MSO and a constant B such that

$$\mathbf{L} = \mathbf{L}(\varphi) \cap \{\mathbf{M} \mid \mathbf{M} \textbf{ is universally B-bounded}\}$$

MSO ...

The proof uses a technique developed by W. Thomas.

MSO ...

The proof uses a technique developed by W. Thomas.

- In one direction interpret the MSO over words formula describing linearizations of L on MSCs.

MSO ...

The proof uses a technique developed by W. Thomas.

- In one direction interpret the MSO over words formula describing linearizations of L on MSCs.
- In the other direction, interpret the MSO over MSCs formula on words. Given φ construct a MSO over words formula that is true of a word w only if

MSO ...

The proof uses a technique developed by W. Thomas.

- In one direction interpret the MSO over words formula describing linearizations of L on MSCs.

- In the other direction, interpret the MSO over MSCs formula on words. Given φ construct a MSO over words formula that is true of a word w only if

    1. w is a linearization of an B-bounded MSC.

MSO ...

The proof uses a technique developed by W. Thomas.

- In one direction interpret the MSO over words formula describing linearizations of L on MSCs.
- In the other direction, interpret the MSO over MSCs formula on words. Given φ construct a MSO over words formula that is true of a word w only if
  1. w is a linearization of an B-bounded MSC.
  2. The MSC M_w satisfies φ. This involves showing that $\leq, <_m$ are definable over w.

MSO ...

The proof uses a technique developed by W. Thomas.

- In one direction interpret the MSO over words formula describing linearizations of L on MSCs.

- In the other direction, interpret the MSO over MSCs formula on words. Given φ construct a MSO over words formula that is true of a word w only if
  1. w is a linearization of an B-bounded MSC.
  2. The MSC M_w satisfies φ. This involves showing that $\leq, <_m$ are definable over w.
  3. M_w is universally B-bounded.

MSO ...

The proof uses a technique developed by W. Thomas.

- In one direction interpret the MSO over words formula describing linearizations of L on MSCs.

- In the other direction, interpret the MSO over MSCs formula on words. Given φ construct a MSO over words formula that is true of a word w only if
  1. w is a linearization of an B-bounded MSC.
  2. The MSC M_w satisfies φ. This involves showing that $\leq, <_m$ are definable over w.
  3. M_w is universally B-bounded.

Observe that $1, 2$ ensure that the set of B-bounded linearizations of $L(\varphi)$ is always a regular language.

MSO ...

The proof uses a technique developed by W. Thomas.

- In one direction interpret the MSO over words formula describing linearizations of L on MSCs.

- In the other direction, interpret the MSO over MSCs formula on words. Given φ construct a MSO over words formula that is true of a word w only if
  1. w is a linearization of an B-bounded MSC.
  2. The MSC M_w satisfies φ. This involves showing that $\leq, <_m$ are definable over w.
  3. M_w is universally B-bounded.

Observe that $1, 2$ ensure that the set of B-bounded linearizations of $L(\varphi)$ is always a regular language.

Theorem: Model-checking MSGs w.r.t. MSO is decidable.

MSO ...

MSO ...

The same observation also leads to the decidability of satisfiability for MSO.

Theorem: Satisfiability is decidable for MSO over the class of universally (existentially) B-bounded models.

MSO ...

The same observation also leads to the decidability of satisfiability for MSO.

Theorem: Satisfiability is decidable for MSO over the class of universally (existentially) B-bounded models.

Further, MSO is strictly more expressive than MPAS w.r.t. general MSCs.

MSO ...

The same observation also leads to the decidability of satisfiability for MSO.

Theorem: Satisfiability is decidable for MSO over the class of universally (existentially) B-bounded models.

Further, MSO is strictly more expressive than MPAS w.r.t. general MSCs.

Theorem: The quantifier alternation hierarchy for MSO over MSCs is strict. In particular EMSO is strictly weaker than MSO.

Existentially bounded languages

Existentially bounded languages

- The model checking problem for MSGs is decidable because the language of an MSG always has a regular set of representatives.

Existentially bounded languages

- The model checking problem for MSGs is decidable because the language of an MSG always has a regular set of representatives.
- If an MSC language has a regular set of representatives then it is existentially bounded.

Existentially bounded languages

- The model checking problem for MSGs is decidable because the language of an MSG always has a regular set of representatives.
- If an MSC language has a regular set of representatives then it is existentially bounded.

Theorem: Let L be an existentially B-bounded language. Then the following statements are equivalent:

1. B-bounded linearizations of L form a regular set of representatives for L

Existentially bounded languages

- The model checking problem for MSGs is decidable because the language of an MSG always has a regular set of representatives.
- If an MSC language has a regular set of representatives then it is existentially bounded.

Theorem: Let L be an existentially B-bounded language. Then the following statements are equivalent:

1. B-bounded linearizations of L form a regular set of representatives for L
2. L is MPA recognisable (with auxiliary messages).

Existentially bounded languages

- The model checking problem for MSGs is decidable because the language of an MSG always has a regular set of representatives.
- If an MSC language has a regular set of representatives then it is existentially bounded.

Theorem: Let L be an existentially B-bounded language. Then the following statements are equivalent:

1. B-bounded linearizations of L form a regular set of representatives for L
2. L is MPA recognisable (with auxiliary messages).
3. L is the MSO definable.

However, deterministic MPAs do not suffice.

Adding time to MSCs

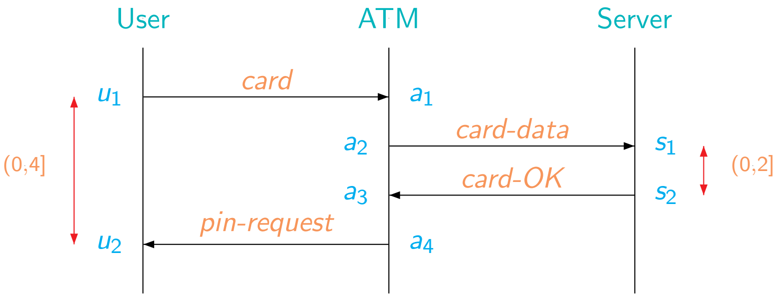
Adding time to MSCs

- Time constrained MSCs
  - MSCs with timing constraints between events

Adding time to MSCs

- Time constrained MSCs
  - MSCs with timing constraints between events
- Time constrained Message Sequence Graphs
  - Generate infinite families of time constrained MSCs

MSCs with time constraints



Time Constrained MSCs

- Associate time interval constraints with pairs of events

Time Constrained MSCs

- Associate time interval constraints with pairs of events
- If $(e, e') \mapsto [l, u]$, then the time between occurrence of e and e' must be between l and u

Time Constrained MSCs

- Associate time interval constraints with pairs of events
- If $(e, e') \longmapsto [l, u]$, then the time between occurrence of e and e' must be between l and u
- Intervals may be open, closed, half-open

Time Constrained MSCs

- Associate time interval constraints with pairs of events
- If $(e, e') \longmapsto [l, u]$, then the time between occurrence of e and e' must be between l and u
- Intervals may be open, closed, half-open
- Simplifying assumptions

Time Constrained MSCs

- Associate time interval constraints with pairs of events
- If $(e, e') \longmapsto [l, u]$, then the time between occurrence of e and e' must be between l and u
- Intervals may be open, closed, half-open
- Simplifying assumptions
  - Interval constraints are local to a process ...

Time Constrained MSCs

- Associate time interval constraints with pairs of events
- If $(e, e') \longmapsto [l, u]$, then the time between occurrence of e and e' must be between l and u
- Intervals may be open, closed, half-open
- Simplifying assumptions
  - Interval constraints are local to a process ...
    - Both e and e' lie on same process line

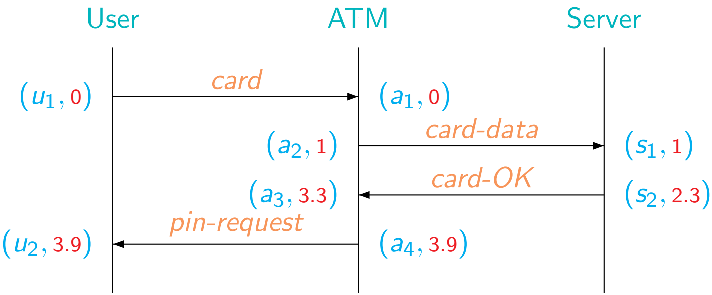
Time Constrained MSCs

- Associate time interval constraints with pairs of events
- If $(e, e') \mapsto [l, u]$, then the time between occurrence of e and e' must be between l and u
- Intervals may be open, closed, half-open
- Simplifying assumptions
  - Interval constraints are local to a process ...
    - Both e and e' lie on same process line
  - ... or across a single message

Time Constrained MSCs

- Associate time interval constraints with pairs of events
- If $(e, e') \mapsto [l, u]$, then the time between occurrence of e and e' must be between l and u
- Intervals may be open, closed, half-open
- Simplifying assumptions
  - Interval constraints are local to a process ...
    - Both e and e' lie on same process line
  - ... or across a single message
    - e is $p!q(m)$ and e' is corresponding receive $q?p(m)$

A timed behaviour



Timed MSCs

- Add timestamps to events on MSC, $\tau : E \to \mathbb{R}_{\geq 0}$

Timed MSCs

- Add timestamps to events on MSC, $\tau : E \rightarrow \mathbb{R}_{\geq 0}$
- All timestamps refer to same global time

Timed MSCs

- Add timestamps to events on MSC, $\tau : E \to \mathbb{R}_{\geq 0}$
- All timestamps refer to same global time
- Order of timestamps respects partial order on events

Timed MSCs

- Add timestamps to events on MSC, $\tau : E \rightarrow \mathbb{R}_{\geq 0}$
- All timestamps refer to same global time
- Order of timestamps respects partial order on events

- Linearizations of timed MSCs are timed words

Timed MSCs

- Add timestamps to events on MSC, $\tau : E \rightarrow \mathbb{R}_{\geq 0}$
- All timestamps refer to same global time
- Order of timestamps respects partial order on events

- Linearizations of timed MSCs are timed words
- Again, a single linearization suffices to reconstruct a timed MSC

Timed MSCs

- Add timestamps to events on MSC, $\tau : E \to \mathbb{R}_{\geq 0}$
- All timestamps refer to same global time
- Order of timestamps respects partial order on events

- Linearizations of timed MSCs are timed words
- Again, a single linearization suffices to reconstruct a timed MSC

- A timed MSC covers a TC-MSC if for each constraint $(e, e') \mapsto [l, u]$, $l \leq \tau(e') - \tau(e) \leq u$

Timed MSCs

- Add timestamps to events on MSC, $\tau : E \to \mathbb{R}_{\geq 0}$
- All timestamps refer to same global time
- Order of timestamps respects partial order on events

- Linearizations of timed MSCs are timed words
- Again, a single linearization suffices to reconstruct a timed MSC

- A timed MSC covers a TC-MSC if for each constraint $(e, e') \mapsto [l, u]$, $l \leq \tau(e') - \tau(e) \leq u$
  - Replace $\leq$ by $<$, as appropriate, for open, half-open intervals

Timed MSCs

- Add timestamps to events on MSC, $\tau : E \to \mathbb{R}_{\geq 0}$
- All timestamps refer to same global time
- Order of timestamps respects partial order on events

- Linearizations of timed MSCs are timed words
- Again, a single linearization suffices to reconstruct a timed MSC

- A timed MSC covers a TC-MSC if for each constraint $(e, e') \mapsto [l, u]$, $l \leq \tau(e') - \tau(e) \leq u$
  - Replace $\leq$ by $<$, as appropriate, for open, half-open intervals
- TC-MSC $T \Rightarrow L(T)$, set of timed MSCs that cover T

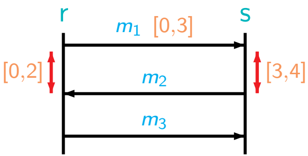
TC-MSCs and Timed MSCs

TC-MSCs and Timed MSCs

- The set of timed MSCs covering a TC-MSC may be empty.

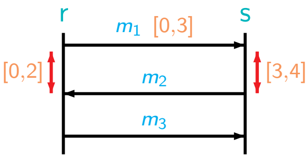
TC-MSCs and Timed MSCs

- The set of timed MSCs covering a TC-MSC may be empty.



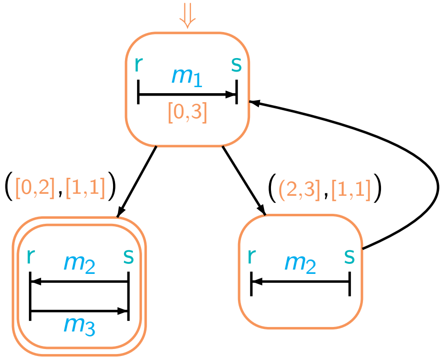
TC-MSCs and Timed MSCs

- The set of timed MSCs covering a TC-MSC may be empty.
- A TC-MSC is said to be realizable if it is covered by atleast one timed MSC.



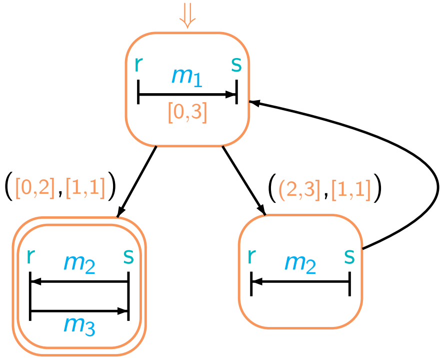
Time Constrained Message Sequence Graphs

- States labelled by time constrained MSCs
- Local constraints for each process along edges
- Legal paths in the automaton generate time constrained MSCs



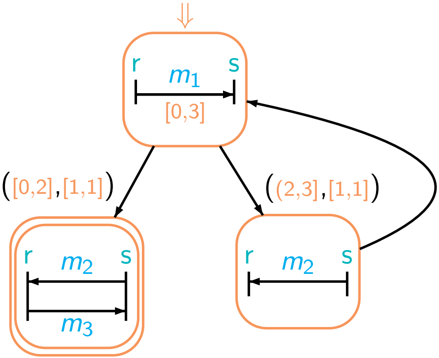
Time Constrained Message Sequence Graphs

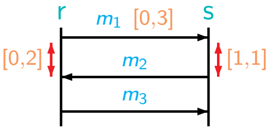
- States labelled by time constrained MSCs
- Local constraints for each process along edges
- Legal paths in the automaton generate time constrained MSCs



Time Constrained Message Sequence Graphs

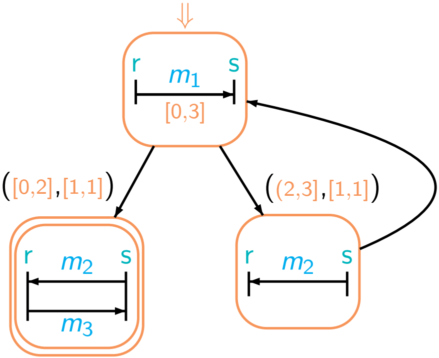
- States labelled by time constrained MSCs
- Local constraints for each process along edges
- Legal paths in the automaton generate time constrained MSCs

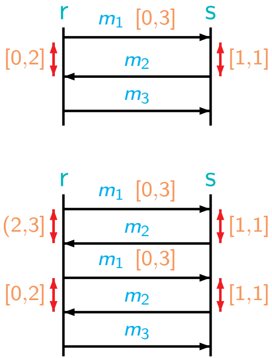




Time Constrained Message Sequence Graphs

- States labelled by time constrained MSCs
- Local constraints for each process along edges
- Legal paths in the automaton generate time constrained MSCs





Reachability

Reachability

Given a TC-MSG G and a state q in G, does there exist a path $q_0 q_1 \ldots q_k = q$ from an initial state q_0 such that the TC-MSG generated by this path is realizable ?

Reachability

Given a TC-MSG G and a state q in G, does there exist a path $q_0 q_1 \dots q_k = q$ from an initial state q_0 such that the TC-MSG generated by this path is realizable ?

(The control state reachability problem for TC-MSGs.)

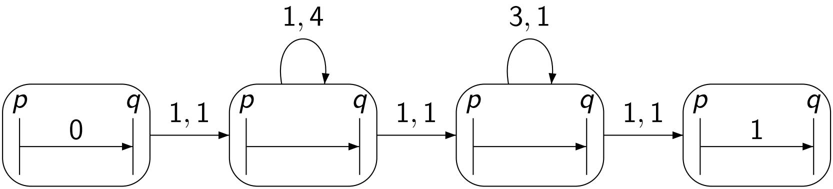
Reachability

Given a TC-MSG G and a state q in G, does there exist a path $q_0 q_1 \dots q_k = q$ from an initial state q_0 such that the TC-MSG generated by this path is realizable ?

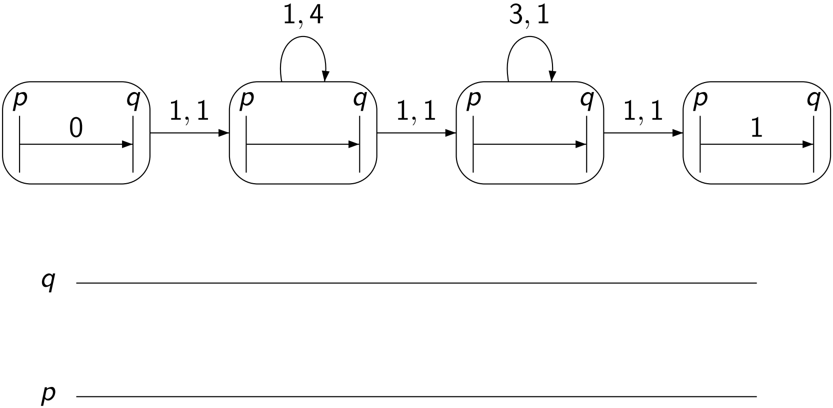
(The control state reachability problem for TC-MSGs.)

This problem is trivial for ordinary MSGs.

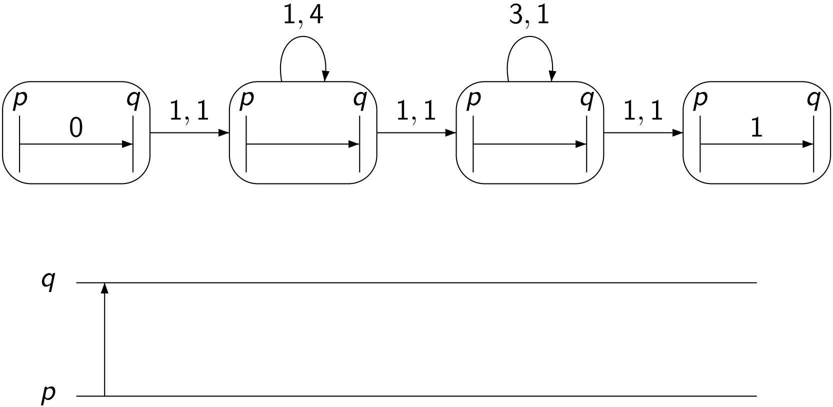
Reachability ...



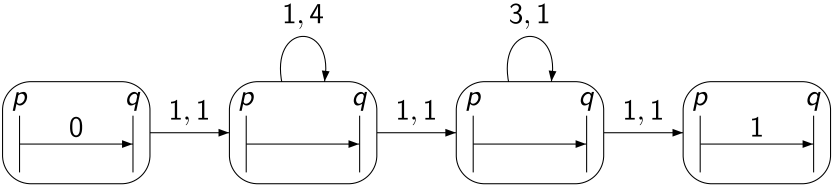
Reachability ...



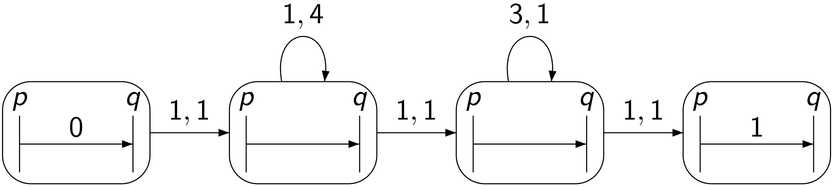
Reachability ...

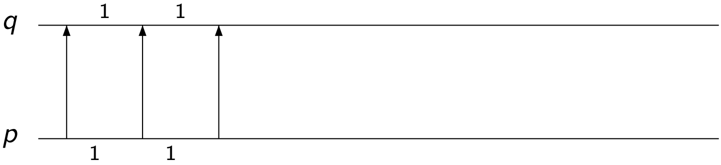


Reachability ...

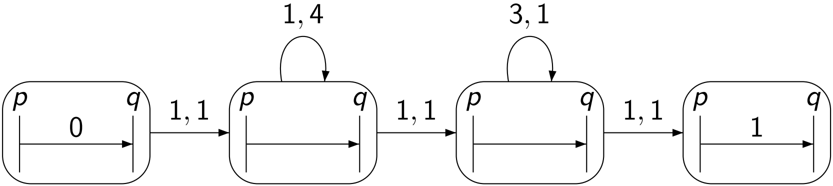


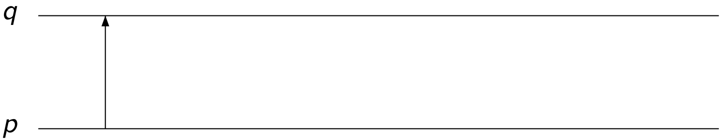
Reachability ...



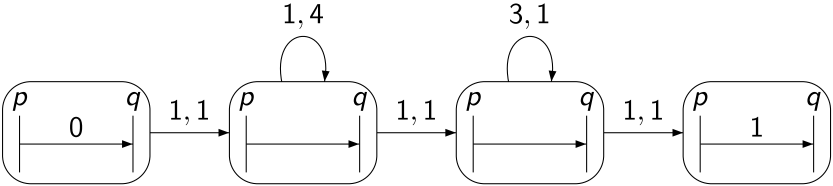


Reachability ...

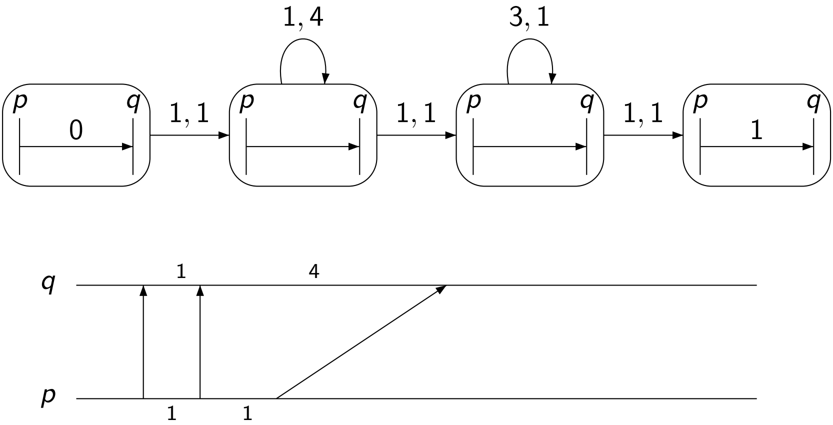




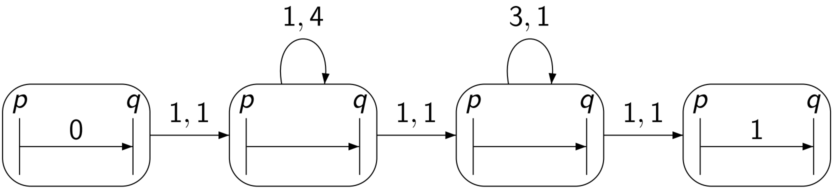
Reachability ...

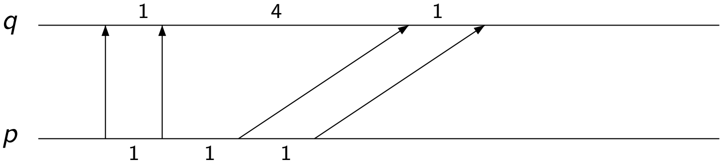


Reachability ...

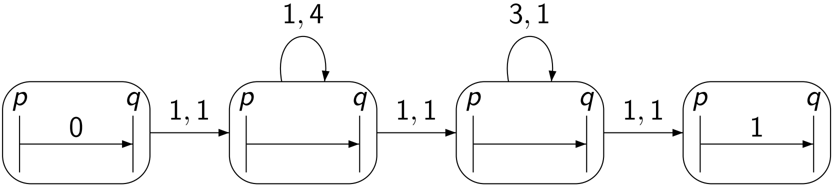


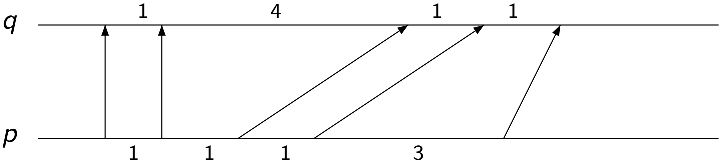
Reachability ...



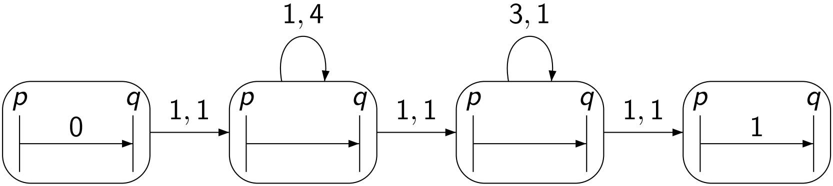


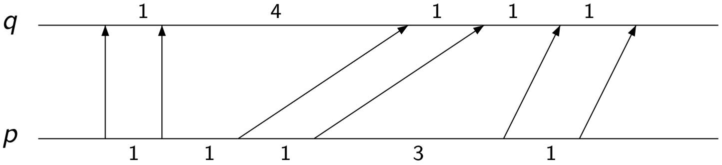
Reachability ...



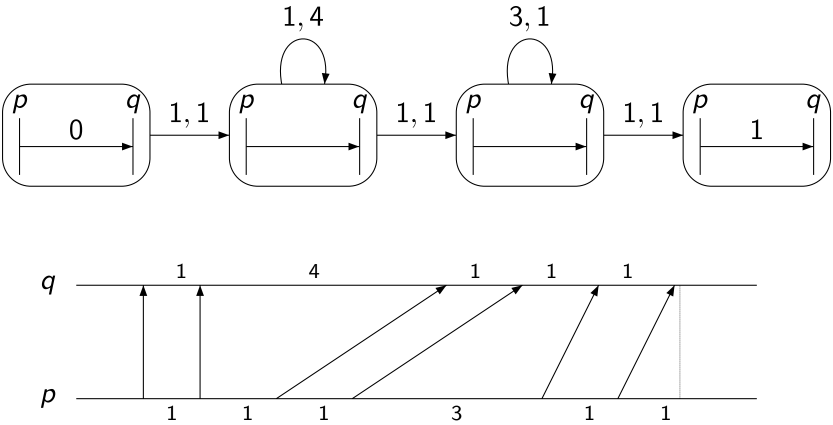


Reachability ...

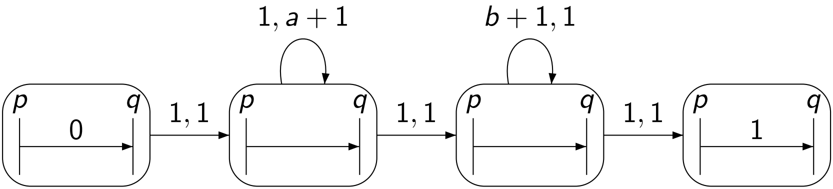




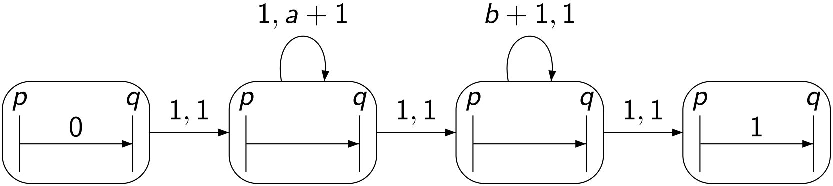
Reachability ...



Reachability ...

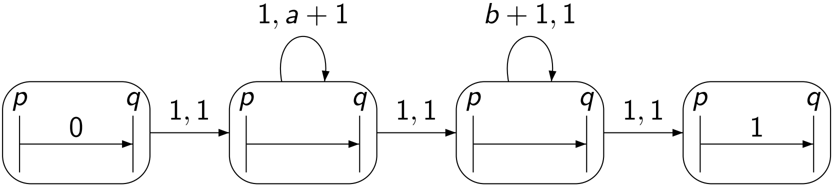


Reachability ...



- The first loop is to be executed k times and the second one l times such that $a.k - b.l = 1$.

Reachability ...



- The first loop is to be executed k times and the second one l times such that $a.k - b.l = 1$.
- Simple paths may not be realizable while those with loops may be.

Boundedness for Timed MSCs

- A timed MSC is universally B bounded if all its timed linearizations are B bounded.

Boundedness for Timed MSCs

- A timed MSC is universally B bounded if all its timed linearizations are B bounded.
- A timed MSC is existentially B bounded if it has at least one timed linearization that is B bounded.

Boundedness for Timed MSCs

- A timed MSC is universally B bounded if all its timed linearizations are B bounded.
- A timed MSC is existentially B bounded if it has at least one timed linearization that is B bounded.
- A TC-MSC is (universally/existentially) B bounded if all its timed realizations are (universally/existentially) B bounded.

Boundedness for Timed MSCs

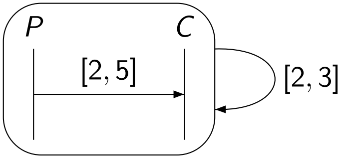
- A timed MSC is universally B bounded if all its timed linearizations are B bounded.
- A timed MSC is existentially B bounded if it has at least one timed linearization that is B bounded.
- A TC-MSC is (universally/existentially) B bounded if all its timed realizations are (universally/existentially) B bounded.
- A TC-MSG is (universally/existentially) bounded if there is a B such that all the TC-MSCs realizing it are (universally/existentially) B bounded.

Boundedness ...

Time constraints may ensure boundedness.

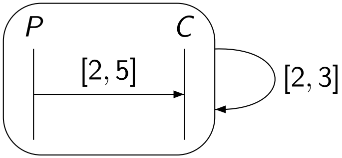
Boundedness ...

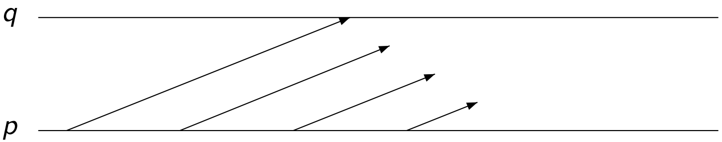
Time constraints may ensure boundedness.



Boundedness ...

Time constraints may ensure boundedness.



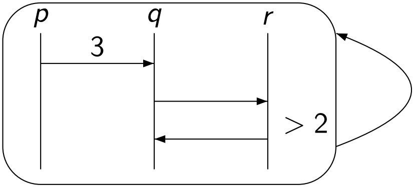


Boundedness ...

Time constraints may ensure boundedness.

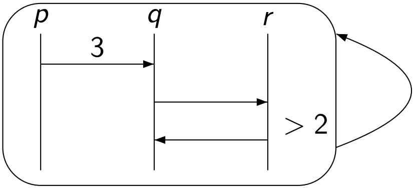
Boundedness ...

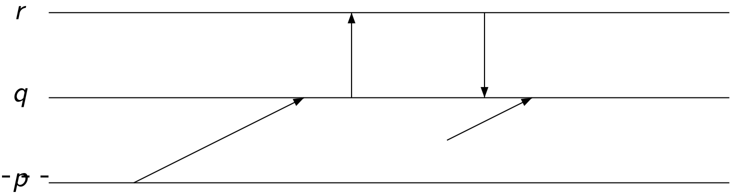
Time constraints may ensure boundedness.



Boundedness ...

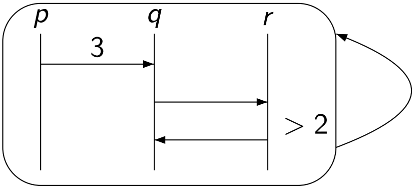
Time constraints may ensure boundedness.

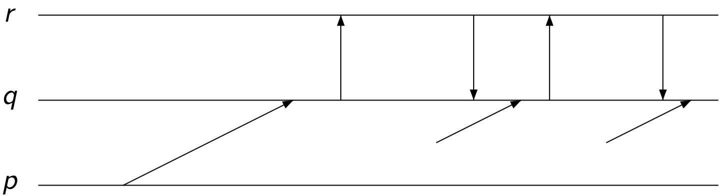




Boundedness ...

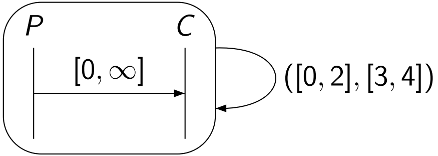
Time constraints may ensure boundedness.





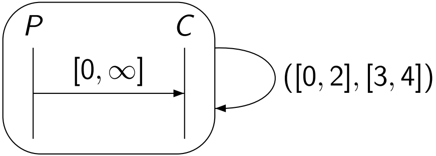
Boundedness ...

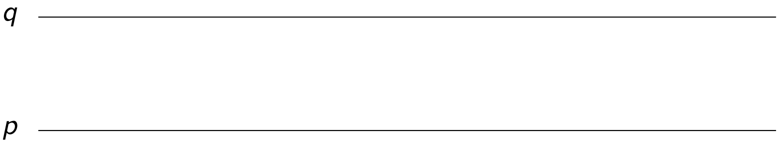
Time contraints may rule out existential boundedness.



Boundedness ...

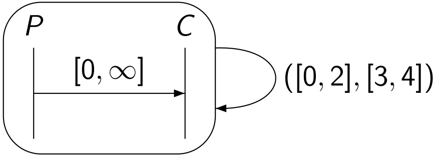
Time contraints may rule out existential boundedness.

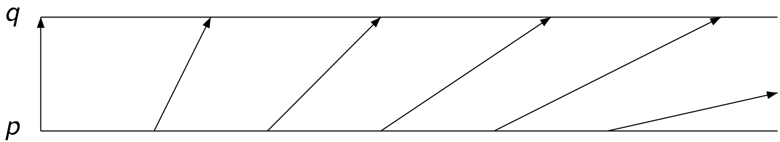




Boundedness ...

Time contraints may rule out existential boundedness.





The Results

Theorem: The control state reachability problem for TC-MSGs is undecidable. The problem is undecidable even when there are no timing constraints on messages.

The Results

Theorem: The control state reachability problem for TC-MSGs is undecidable. The problem is undecidable even when there are no timing constraints on messages.

The (language) emptiness problem for TC-MSGs is undecidable.

The Results

Theorem: The control state reachability problem for TC-MSGs is undecidable. The problem is undecidable even when there are no timing constraints on messages.

The (language) emptiness problem for TC-MSGs is undecidable.

- The problem remains undecidable even if all constraints are open intervals.

The Results

Theorem: The control state reachability problem for TC-MSGs is undecidable. The problem is undecidable even when there are no timing constraints on messages.

The (language) emptiness problem for TC-MSGs is undecidable.

- The problem remains undecidable even if all constraints are open intervals.
- The problem remains undecidable even if all across node constraints are on a single process p.

The Results

Theorem: The control state reachability problem for TC-MSGs is undecidable. The problem is undecidable even when there are no timing constraints on messages.

The (language) emptiness problem for TC-MSGs is undecidable.

- The problem remains undecidable even if all constraints are open intervals.
- The problem remains undecidable even if all across node constraints are on a single process p.
- The reachability problem for locally synchronized TC-MSGs is decidable.

The Results

Theorem: The control state reachability problem for TC-MSGs is undecidable. The problem is undecidable even when there are no timing constraints on messages.

The (language) emptiness problem for TC-MSGs is undecidable.

- The problem remains undecidable even if all constraints are open intervals.
- The problem remains undecidable even if all across node constraints are on a single process p.
- The reachability problem for locally synchronized TC-MSGs is decidable.

Thank you.

Edge Constraint free TC-MSGs

Consider TC-MSGs where there are no time constraints associated with transitions between nodes.

Edge Constraint free TC-MSGs

Consider TC-MSGs where there are no time constraints associated with transitions between nodes.

- The control state reachability problem is decidable. A path is realizable if and only if each node in the path is realizable.

Edge Constraint free TC-MSGs

Consider TC-MSGs where there are no time constraints associated with transitions between nodes.

- The control state reachability problem is decidable. A path is realizable if and only if each node in the path is realizable.
- The boundedness problem is still open. Time constraints can enforce boundedness.