Overview

1 Lecture 1a: Introduction

2 Lecture 1b: Message Sequence Charts
Theoretical Foundations of the UML

Lecture 1a: Introduction

Joost-Pieter Katoen

Lehrstuhl für Informatik 2
Software Modeling and Verification Group

moves.rwth-aachen.de/teaching/ss-20/fuml/

April 20, 2020
Target audience

You are studying:
- Master Computer Science, or
- Master Data Science, or
- Master Systems Software Engineering, or
- Bachelor Computer Science, or
- ......

Usage as:
- elective course Theoretical Computer Science
- not a Wahlpflicht course for bachelor students
- specialization MOVES (Modeling and Verification of Software)
- complementary to Model-based Software Development (Rumpe)
In general:

- interest in system software engineering
- interest in formal methods for software
- interest in semantics and verification
- application of mathematical reasoning

Prerequisites:

- mathematical logic
- formal language and automata theory
- algorithms and data structures
- computability and complexity theory
- regular languages
- finite state automata
- complexity classes (NP, PSPACE)
- undecidability
### People involved:

<table>
<thead>
<tr>
<th>Lecturer</th>
<th>EMail</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lectures: Joost-Pieter Katoen</td>
<td><a href="mailto:katoen@cs.rwth-aachen.de">katoen@cs.rwth-aachen.de</a></td>
</tr>
<tr>
<td>Exercises: Mingshuai Chen</td>
<td><a href="mailto:chenms@cs.rwth-aachen.de">chenms@cs.rwth-aachen.de</a></td>
</tr>
<tr>
<td></td>
<td>Bahare Salmani</td>
</tr>
</tbody>
</table>
Organization

Schedule under the current COVID-19 circumstances:

- The lectures will take place in digital form (slide-casts).
- The exercise classes will take place in digital form (slide-casts).
- A weekly Q&A session (on Thu, 16:00–17:30) via Zoom starting from April 23.
- There will be about 21 lectures and 10 exercise classes.
- Two lecture slide-casts per week starting from April 20.
- One exercise class slide-cast per week starting from April 27.
Home assignments:

- **weekly assignments**: about 4 exercises to be solved by you
- groups of maximally three students together work on assignments
- solutions: hand in via RWTHmoodle\(^a\) as pdf-file
- first assignment: **Monday April 20**
- solution due at start next week: **Monday April 27, 09:00**
- first on-line exercise class video: **Monday April 27**
- this scheme is repeated on a weekly basis until the beginning of July
- no lecture+exercise class in week following Pentecost

\(^a\)You get access by enrolling to the exercise class via RWTHonline.
Examination: (6 ECTS credit points)

- written exam: July 23, 2020, 13:30-15:30 (Aula 2)

Details

- Admission: at least 40% of total amount of exercise points
- Registration: between May 1 and July 1 (via RWTHonline).

10 exercise classes of 100 points each

≥ 400 points to be earned
Motivation

Scope:
- **Goal:** formal description + analysis of (concurr.) software systems
- **Focus:** the Unified Modeling Language

More specifically:
- Sequence Diagrams (used for requirements analysis)
- Propositional Dynamic Logic
- Communicating Finite State Automata
- Statecharts (behavioral description of systems)
Motivation

Scope:

- **Goal:** formal description + analysis of (concurr.) software systems
- **Focus:** the Unified Modeling Language

More specifically:

- Sequence Diagrams (used for requirements analysis)
- Propositional Dynamic Logic
- Communicating Finite State Automata
- Statecharts (behavioral description of systems)

Aims:

- clarify and make precise the semantics of some UML fragments
- formal reasoning about basic properties of UML models
- convince you that UML models are much harder than you think
What this course is **NOT** about:

- the use of the UML in the software development cycle
  - see the complementary course by Prof. Rumpe
- other notations of the UML (e.g., class diagrams, activity diagrams)
- what is precisely in the UML, and what is not
  - liberal interpretation of which constructs belong to the UML
- applying the UML to concrete SW development case studies
- empirical results on the usage of UML
- drawing pictures
- ...

Joost-Pieter Katoen
Theoretical Foundations of the UML
1 Lecture 1a: Introduction

2 Lecture 1b: Message Sequence Charts
1970s – 1980s: often used informally

1992: first version of MSCs standardized by CCITT (currently ITU) Z.120

1992 – 1996: many extensions, e.g., high-level + formal semantics (using process algebras)

1996: MSC’96 standard

2000: MSC 2000, time, data, o-o features

2005: MSC 2004

2011: latest standard published
Variants of MSCs

- UML sequence diagrams
- (instantiations of) use cases
- triggered MSCs
- netcharts (= Petri net + MSC)
- STAIRS
- Live sequence charts

...
Characteristics

- scenario-based language
- visual representation
- “easy” to comprehend
- generalization possible towards automata (states are MSCs)
- widely used in industrial practice
Applications

- requirements specification
  (positive, negative scenarios, e.g., CREWS)

- system design and software engineering

- visualization of test cases
  (graphical extension to TTCN)

- feature interaction detection

- workflow management systems

- ...
These pictures are formalized using partial orders.

message: \( \text{Roces} \) \( \text{Ses} \)

\( \vdash \) \text{send event} \( (\text{p}1, \text{a}, \text{p}2) \)

\( \vdash \) \text{event} \( \text{b} \)

\( \vdash \) \text{end at process p}3

\( \vdash \) \text{describes a single scenario}

\( \vdash \) \text{receive} \( ?(\text{p}2, \text{a}, \text{p}1) \)
These pictures are formalized using **partial orders**.
Partial orders

**Definition**

Let $E$ be a set of events. A **partial order** over $E$ is a relation $\leq \subseteq E \times E$ such that:

1. $\leq$ is reflexive, i.e., $\forall e \in E. e \leq e$,
2. $\leq$ is transitive, i.e., $e \leq e' \land e' \leq e''$ implies $e \leq e''$, and
3. $\leq$ is anti-symmetric, i.e., $\forall e, e'. (e \leq e' \land e' \leq e) \Rightarrow e = e'$.

The pair $(E, \leq)$ is called a **partially ordered set** (poset, for short).

**Example**

$E$ : sets of natural numbers  
eg. $e \in E = \{0,2,7\}$

ordering $\leq$ $(E, \leq)$ is poset.

1. for every set $e$, $e \leq e$
2. $e \leq f$ and $f \leq g$ then $e \leq g$
3. $e \leq f$ and $f \leq e$ then $e = f$
\[ E = \text{bitstrings}, \text{ e.g. } e \in E = 01011 \]

\[ e \leq f \ \text{iff} \ \text{length}(e) \leq \text{length}(f) \]

\((E, \leq)\) is not a poset.

1. \(e \leq e\)

2. \(e \leq f \text{ and } f \leq g \text{ then } e \leq g\)

3. \(e \leq f \text{ and } f \leq e \text{ , then }\)

\[ \text{length}(e) \leq \text{length}(f) \text{ and } \]

\[ \text{length}(f) \leq \text{length}(e) \]

but not necessarily \(e = f\).
Partial orders

**Definition**

Let $E$ be a set of events.

A **partial order** over $E$ is a relation $\leq \subseteq E \times E$ such that:

1. $\leq$ is reflexive, i.e., $\forall e \in E. e \leq e$,
2. $\leq$ is transitive, i.e., $e \leq e' \land e' \leq e''$ implies $e \leq e''$, and
3. $\leq$ is anti-symmetric, i.e., $\forall e, e'. (e \leq e' \land e' \leq e) \Rightarrow e = e'$.

The pair $(E, \leq)$ is called a **partially ordered set** (poset, for short).

**Definition**

Let $(E, \leq)$ be a poset and let $e, e' \in E$. $e$ and $e'$ are **comparable** if $e \leq e'$ or $e' \leq e$. Otherwise, they are **incomparable**.

The set $\{1,2\}$ is incomparable to $\{3\}$, neither $\{3\} \leq \{1,2\}$ nor $\{1,2\} \leq \{3\}$. 
Partial orders

Definition

Let $E$ be a set of events.
A partial order over $E$ is a relation $\leq \subseteq E \times E$ such that:

1. $\leq$ is reflexive, i.e., $\forall e \in E. e \leq e$,
2. $\leq$ is transitive, i.e., $e \leq e' \land e' \leq e''$ implies $e \leq e''$, and
3. $\leq$ is anti-symmetric, i.e., $\forall e, e'. (e \leq e' \land e' \leq e) \Rightarrow e = e'$.

The pair $(E, \leq)$ is called a partially ordered set (poset, for short).

Definition

Let $(E, \leq)$ be a poset and let $e, e' \in E$. $e$ and $e'$ are comparable if $e \leq e'$ or $e' \leq e$. Otherwise, they are incomparable.

$\leq$ is a non-strict partial order as it is reflexive. A strict partial order is a relation $<$ that is irreflexive, transitive and asymmetric (i.e., if $e < e'$ then not $e' < e$).
Definition

Let \((E, \leq)\) be a poset.

The Hasse diagram \((E, \preceq)\) of \((E, \leq)\) is defined by:

\[
e \preceq e' \text{ iff } e \leq e' \text{ and } \neg(\exists e'' \neq e, e'. e \leq e'' \wedge e'' \leq e')
\]

Hasse diagrams can be used to visualize posets with finitely many elements in a succinct way.
Let \((E, \leq)\) be a poset. A **linearization** of \((E, \leq)\) is a total order \(\leq \subseteq E \times E\) such that

\[ e \leq e' \quad \text{implies} \quad e \sqsubseteq e' \]

A linearization is a topological sort of the Hasse diagram of \((E, \leq)\).

Note that every partial order has at least one linearization.
Example

Let \( E = \{e_1, \ldots, e_6\} \),

\[ \leq = \{(e_1, e_2), (e_1, e_3), (e_3, e_4), (e_4, e_5), (e_1, e_6), (e_3, e_5), (e_1, e_5), (e_5, e_6), (e_1, e_4), (e_3, e_6), (e_4, e_6)\}^r \]

where \( R^r \) denotes the reflexive closure of \( R \)

Hasse diagram:

\[ e_1 \rightarrow e_2 \rightarrow e_3 \rightarrow e_4 \rightarrow e_5 \rightarrow e_6 \]

Linearizations:
- \( e_1 e_2 e_3 e_4 e_5 e_6 \),
- \( e_1 e_3 e_2 e_4 e_5 e_6 \),
- \( e_1 e_3 e_4 e_2 e_5 e_6 \),
- \( e_1 e_3 e_4 e_5 e_2 e_6 \),
- \( e_1 e_3 e_4 e_5 e_6 e_2 \)

No linearizations:
- \( e_2 e_1 e_3 \ldots \), and \( e_1 e_4 e_3 \ldots \)
Processes and actions

Definition

Let $\mathcal{P}$: finite set of (sequential) processes
$\mathcal{C}$: finite set of message contents ($a, b, c, \ldots \in \mathcal{C}$)

Communication action:
$p, q \in \mathcal{P}$, $p \neq q$, $a \in \mathcal{C}$
$(!) (p, q, a)$ "process $p$ sends message $a$ to process $q$"$
?(p, q, a)$ "process $p$ receives message $a$ sent by process $q$"
Processes and actions

**Definition**

Let \( \mathcal{P} \): finite set of (sequential) processes
\( \mathcal{C} \): finite set of message contents \((a, b, c, \ldots \in \mathcal{C})\)

**Definition**

Communication action: \( p, q \in \mathcal{P}, p \neq q, a \in \mathcal{C} \)

\![p, q, a] \quad \text{“process } p \text{ sends message } a \text{ to process } q”

\?[p, q, a] \quad \text{“process } p \text{ receives message } a \text{ sent by process } q”

Let \( \text{Act} \) denote the set of communication actions \((\text{over } \mathcal{P} \text{ and } \mathcal{C})\)
An MSC $M = (\mathcal{P}, E, C, l, m, \preceq)$ with:

- $\mathcal{P}$, a finite set of processes $\{p_1, p_2, \ldots, p_n\}$
- $E$, a finite set of events $E = E \cap \mathcal{P}$
- $C$, a finite set of message contents $l : E \to \text{Act}$, a labelling function defined by:
  
  $l(e) = \begin{cases} 
  (\!, p, q, a) & \text{if } e \in E \setminus \mathcal{P} \setminus E? \ \\ 
  (?, p, q, a) & \text{if } e \in E \setminus \mathcal{P} \setminus E!. 
  \end{cases}$
  
  for $p \neq q \in \mathcal{P}$, $a \in C$.
Message Sequence Chart (MSC) (1)

Definition

An MSC $M = (\mathcal{P}, E, C, l, m, \leq)$ with:

- $\mathcal{P}$, a finite set of processes $\{p_1, p_2, \ldots, p_n\}$ with $n > 1$
- $E$, a finite set of events

$$E = \bigcup_{p \in \mathcal{P}} E_p = E? \uplus E!$$

- $E_{p_1} = \{e, e'\}$
- $e \in E_1$
- $e' \in E_2$
- $E_{p_2} = \{f, f', f''\}$
- $f \in E_2$
- $f' \in E_2$
- $f'' \in E_2$
Definition

An MSC \( M = (\mathcal{P}, E, C, l, m, \preceq) \) with:

- \( \mathcal{P} \), a finite set of processes \( \{p_1, p_2, \ldots, p_n\} \) with \( n > 1 \)
- \( E \), a finite set of events

\[
E = \bigcup_{p \in \mathcal{P}} E_p = E_? \uplus E_!
\]

- \( C \), a finite set of message contents
- \( l : E \to Act \), a labelling function defined by:

\[
l(e) = \begin{cases} 
!(p, q, a) & \text{if } e \in E_p \cap E_! \\
?p(p, q, a) & \text{if } e \in E_p \cap E_? 
\end{cases}
\]

\( \text{for } p \neq q \in \mathcal{P}, a \in C \)
Definition

\( m : E! \to E? \) a bijection ("matching function"), satisfying:

\[
\begin{align*}
    m(e) &= e' \\
    l(e) &= !(p, q, a) \implies l(e') &= ?(q, p, a) \\
    (p \neq q, a \in C)
\end{align*}
\]

\( m(e) = e' \) and corresponding receive event of \( !(p, q, a) \)
**Definition**

- \( m : E! \rightarrow E? \) a bijection ("matching function"), satisfying:

\[
m(e) = e' \land l(e) = !(p, q, a) \implies l(e') = ?(q, p, a) \quad (p \neq q, \ a \in \mathcal{C})
\]

- \( \preceq \subseteq E \times E \) is a partial order ("visual order") defined by:

\[
\preceq = \left( \bigcup_{p \in \mathcal{P}} <_p \right) \cup \{(e, m(e)) \mid e \in E_!\}
\]

where for relation \( R \), \( R^* \) denotes its reflexive and transitive closure.

\( <_p \) is a total order = "top-to-bottom" order on process \( p \)

communication order \( <_c \)
Example (1)

\[ M = (\mathcal{P}, E, \mathcal{C}, l, m, \preceq) \] with:
- \[ \mathcal{P} = \{p_1, p_2\} \]
- \[ E_{p_1} = \{e_1, e_4\} \]
- \[ E_{p_2} = \{e_2, e_3\} \]
- \[ \mathcal{C} = \{a, b\} \]
- \[ E_! = \{e_1, e_3\} \], \[ E_? = \{e_2, e_4\} \]
- \[ l(e_1) = !(p_1, p_2, a) \]
- \[ l(e_2) = ?(p_2, p_1, a) \]
- \[ l(e_3) = !(p_2, p_1, b) \]
- \[ l(e_4) = ?(p_1, p_2, b) \]
- \[ m(e_1) = e_2 \]
- \[ m(e_3) = e_4 \]

Define \[ \preceq : \]

Ordering at processes: \( e_1 \preceq_{p_1} e_4 \) and \( e_2 \preceq_{p_2} e_3 \).

Hasse diagram of \((E, \preceq)\):

\[ e_1 \rightarrow e_2 \rightarrow e_3 \rightarrow e_4 \]

Linearizations?

\[ \{e_1, e_2, e_3, e_4\} \]}
Example (2)

MSC

\[ M' = (\mathcal{P}, E, C, l, m, \preceq') \] with:

- \( m(e_1) = e_2 \)
- \( m(e_3) = e_4 \)

Hasse diagram of \( \preceq' \):

1. \( \leq'_c : \begin{align*}
    e_1 & \rightarrow e_2 \\
    e_3 & \rightarrow e_4
\end{align*} \)

2. \( \leq'_{p_1} : \begin{align*}
    e_1 & \rightarrow e_4 \\
    e_3 & \rightarrow e_2
\end{align*} \)

3. \( \leq'_{p_2} : \begin{align*}
    e_3 & \rightarrow e_2
\end{align*} \)
This is not an MSC

MSC

\[ p_1 \rightarrow a \rightarrow p_2 \]

\[ b \]

Joost-Pieter Katoen
Theoretical Foundations of the UML
MSC $M = (P, E, C, l, m, \preceq)$ has the *First-In-First-Out* (FIFO) property whenever: for all $e, e' \in E$ we have

$$e < e' \land l(e) = !(p, q, a) \land l(e') = !(p, q, b) \text{ implies } m(e) \prec m(e')$$

i.e., “no message overtaking allowed”

\[
\begin{align*}
e \leq e' & \land e \neq e' \\
\end{align*}
\]
MSC \( M = (\mathcal{P}, E, \mathcal{C}, l, m, \preceq) \) has the First-In-First-Out (FIFO) property whenever: for all \( e, e' \in E \) we have

\[
e < e' \land l(e) = !(p, q, a) \land l(e') = !(p, q, b) \implies m(e) < m(e')
\]

i.e., “no message overtaking allowed”
MSC $M = (\mathcal{P}, E, C, l, m, \preceq)$ has the First-In-First-Out (FIFO) property whenever: for all $e, e' \in E$ we have

$$e < e' \land l(e) = !(p, q, a) \land l(e') = !(p, q, b) \text{ implies } m(e) \prec m(e')$$

i.e., “no message overtaking allowed”

**Note:**
We assume an MSC to possess the FIFO property, unless stated otherwise!
Definition

Let $Lin(M)$ denote the set of (action) linearizations of MSC $M$. 

\[ Lin(M) = \{ !((p,a,o),(a,p,o)) \} \]

\[ ee' \rightarrow x(e) r(e') \]

\[ = !((p,a,o),(a,p,o)) \]
Let $\text{Lin}(M) =$ denote the set of (action) linearizations of MSC $M$.

$\text{Lin}(M)$ denotes a set of words over actions (and not over events)
the word of linearization $e_1 \ldots e_n$ equals $\ell(e_1) \ldots \ell(e_n)$
MSCs and its linearizations are interchangeable

There is a one-to-one correspondence between an MSC and its set of linearizations.

MSC $M$ \Rightarrow Lin(M)
MSCs and its linearizations are interchangeable

There is a one-to-one correspondence between an MSC and its set of linearizations.

We will establish: the set $\text{Lin}(M)$ uniquely characterizes the MSC $M$ (up to the event identities).

From MSCs to its set of linearizations is straightforward.

The reverse direction is discussed in the following. First: well-formedness.

$\omega = \text{!}(p,q,a) ?(q,p,a) \ldots \ldots$
Well-formedness

Let $Ch := \{(p, q) \mid p \neq q, p, q \in \mathcal{P}\}$ be the set of channels over $\mathcal{P}$.

We call $w = a_1 \ldots a_n \in Act^*$ proper if

$!(p, q, a) \ ?(q, p, a)$
Well-formedness

Let \( Ch := \{(p, q) \mid p \neq q, p, q \in \mathcal{P}\} \) be the set of channels over \( \mathcal{P} \).

We call \( w = a_1 \ldots a_n \in Act^* \) proper if every receive in \( w \) is preceded by a corresponding send, i.e.:

\[
\forall (p, q) \in Ch \text{ and prefix } u \text{ of } w, \text{ we have:}
\]

\[
\sum_{m \in C} |u|!(p,q,m) \geq \sum_{m \in C} |u|?(q,p,m)
\]

\( \# \) sends from \( p \) to \( q \) \quad \# \) receipts by \( q \) from \( p \)

where \( |u|_a \) denotes the number of occurrences of action \( a \) in \( u \).

Forbids \( w = ?(q,p,a) ! (p,q,a) \) \( w = a_1 \ldots a_i \ldots a_k \)
Well-formedness

Let \( Ch := \{(p, q) \mid p \neq q, p, q \in \mathcal{P}\} \) be the set of channels over \( \mathcal{P} \).

We call \( w = a_1 \ldots a_n \in \text{Act}^* \) proper if

1. every receive in \( w \) is preceded by a corresponding send, i.e.: for every \( (p, q) \in Ch \) and prefix \( u \) of \( w \), we have:

\[
\sum_{m \in \mathcal{C}} |u|!(p,q,m) \geq \sum_{m \in \mathcal{C}} |u|?(q,p,m)
\]

where \( |u|_a \) denotes the number of occurrences of action \( a \) in \( u \)

2. the FIFO policy is respected, i.e.: for every \( 1 \leq i < j \leq n \), \( (p, q) \in Ch \), and \( a_i = !(p,q,m_1) \), \( a_j = ?(q,p,m_2) \):

\[
\sum_{m \in \mathcal{C}} |a_1 \ldots a_{i-1}|!(p,q,m) = \sum_{m \in \mathcal{C}} |a_1 \ldots a_{j-1}|?(q,p,m) \quad \text{implies} \quad m_1 = m_2
\]

forbids, e.g., \( w = !(p,q,a) !(p,q,b) ?(q,p,b) ?(q,p,a) \).
Well-formedness

Let \( Ch := \{(p, q) \mid p \neq q, p, q \in \mathcal{P}\} \) be the set of channels over \( \mathcal{P} \).

We call \( w = a_1 \ldots a_n \in Act^* \) proper if

1. every receive in \( w \) is preceded by a corresponding send, i.e.:
   \[ \forall (p, q) \in Ch \text{ and prefix } u \text{ of } w, \text{ we have:} \]
   \[
   \sum_{m \in \mathcal{C}} |u|!(p,q,m) \geq \sum_{m \in \mathcal{C}} |u|?(q,p,m)
   \]
   \# sends from \( p \) to \( q \) \hspace{1cm} \# receipts by \( q \) from \( p \)

   where \( |u|_a \) denotes the number of occurrences of action \( a \) in \( u \)

2. the FIFO policy is respected, i.e.:
   \[ \forall 1 \leq i < j \leq n, (p, q) \in Ch, \text{ and } a_i = !(p, q, m_1), a_j = ?(q, p, m_2): \]
   \[
   \sum_{m \in \mathcal{C}} |a_1 \ldots a_{i-1}|!(p,q,m) = \sum_{m \in \mathcal{C}} |a_1 \ldots a_{j-1}|?(q,p,m) \]
   implies \( m_1 = m_2 \)

A proper word \( w \) is well-formed if

\[
\sum_{m \in \mathcal{C}} |w|!(p,q,m) = \sum_{m \in \mathcal{C}} |w|?(q,p,m)
\]
\[
\begin{aligned}
w &= \underbrace{! (r, g, m_1) \ldots ! (p, g, m_2) ! (p, a, m_2)}_{\text{2}} \ldots \underbrace{? (a, p, m_2) ? (a, p, m_3)}_{\text{1}} \\
\text{Claim: } w & \text{ is well-formed.} \\
&= w \text{ is proper + no "pending" messages}
\end{aligned}
\]

1. Consider e.g., prefix \( u = ! (r, g, m) \ldots ? (a, p, m) \):

\[
\sum |u| ! (p, g, m) \geq \sum |u| ? (a, p, m)
\]

also
\[
\sum |u| ! (r, g, m) \geq \sum |u| ? (a, r, m)
\]

check for all prefixes \( u \) of \( w \).

2. FIFO property:

For \((p, g)\): \( m_1 \) is sent and received before \( m_2 \)

\((r, g)\): initially, as there is only 1 message

3. Every sent message in \( w \) is also received in \( w \).
Proportion

For every MSC $M$ and every $w \in Lin(M)$, $w$ is well-formed.
Lemma \( \forall \text{MSC } M. \ \forall w \in \text{Lin}(M) \),
\( w \) is well-formed.

Proof: let \( w = a_1, \ldots, a_n \in \text{Lin}(M) \)

1. by definition if \( a_j = ?(a, p, m) \) then
   \( \exists a_i \) with \( i < j \) \( a_i = !(p, q, m) \). As this holds for every \( j \) it follows that for every prefix \( u \) of \( w \) and every \((p, q)\) it holds:
   \[
   \sum \limits_{m} |u| !(p, q, m) > \sum \limits_{m} |w| ?(a, p, m)
   \]

2. by definition \( M \) is FIFO, thus \( w \) respects the FIFO property.

3. for every \( a_j = !(p, q, m) \), \( w \) contains a corresponding receive event \( a_j = ?(a, p, m) \).

as \( w \) is a linearization of \( \text{MSC } M \).
From linearizations to MSCs

\[ \text{Lin}(M) \longrightarrow M \]
From linearizations to MSCs

Associate to $w = a_1 \ldots a_n \in \text{Act}^*$ an \textit{Act}-labelled poset

$$M(w) = (E, \preceq, \ell)$$

Example construct $M(w)$ for $w = ! (p,q,m) ! (p,q,m_1) ! (p,q,m_2) ? (q,p,m_1) ? (q,p,m_2) ? (q,r,m)$
From linearizations to MSCs

Associate to \( w = a_1 \ldots a_n \in Act^* \) an \( Act \)-labelled poset

\[ M(w) = (E, \preceq, \ell) \]

such that:

- \( E = \{1, \ldots, n\} \) are the positions in \( w \) labelled with \( \ell(i) = a_i \)
Associate to \( w = a_1 \ldots a_n \in \text{Act}^* \) an \textit{Act}-labelled poset \( M(w) = (E, \preceq, \ell) \) such that:

- \( E = \{1, \ldots, n\} \) are the positions in \( w \) labelled with \( \ell(i) = a_i \)
- \( \preceq = \left( \bigcup_{p \in P} \prec_p \cup \prec_{\text{msg}} \right)^* \) where
  - \( i \prec_p j \) if and only if \( i < j \), for every \( i, j \in E_p \)

\[ \uparrow \]

ordering on \( \mathbb{N} \)
From linearizations to MSCs

Associate to \( w = a_1 \ldots a_n \in Act^* \) an \( Act \)-labelled poset

\[
M(w) = (E, \preceq, \ell)
\]
such that:

- \( E = \{1, \ldots, n\} \) are the positions in \( w \) labelled with \( \ell(i) = a_i \)
- \( \preceq = \left( \bigcup_{p \in P} \prec_p \cup \prec_{\text{msg}} \right)^* \) where
  - \( i \prec_p j \) if and only if \( i < j \), for every \( i, j \in E_p \)
  - \( i \prec_{\text{msg}} j \) if for some \( (p, q) \in \text{Ch} \) and \( m \in \mathcal{C} \) we have:

\[
\ell(i) = !(p, q, m) \quad \text{and} \quad \ell(j) = ?(q, p, m)
\]

\[
\sum_{m \in \mathcal{C}} |a_1 \ldots a_{i-1}|!(p, q, m) = \sum_{m \in \mathcal{C}} |a_1 \ldots a_{j-1}|?(q, p, m)
\]
From linearizations to MSCs

Associate to $w = a_1 \ldots a_n \in \text{Act}^*$ an \textit{Act}-labelled poset

$$M(w) = (E, \preceq, \ell)$$

such that:

- $E = \{1, \ldots, n\}$ are the positions in $w$ labelled with $\ell(i) = a_i$
- $\preceq = \left( \bigcup_{p \in \mathcal{P}} \prec_p \cup \prec_{\text{msg}} \right)^*$ where
  - $i \prec_p j$ if and only if $i < j$, for every $i, j \in E_p$
  - $i \prec_{\text{msg}} j$ if for some $(p, q) \in \text{Ch}$ and $m \in \mathcal{C}$ we have:

$$\ell(i) = !(p, q, m) \text{ and } \ell(j) = ?(q, p, m) \text{ and } \sum_{m \in \mathcal{C}} |a_1 \ldots a_{i-1}!((p,q,m)} = \sum_{m \in \mathcal{C}} |a_1 \ldots a_{j-1}?((q,p,m)}$$
From linearizations to MSCs

Associate to $w = a_1 \ldots a_n \in Act^*$ an $Act$-labelled poset

$$M(w) = (E, \preceq, \ell)$$

such that:

- $E = \{1, \ldots, n\}$ are the positions in $w$ labelled with $\ell(i) = a_i$
- $\preceq = \left( \bigcup_{p \in P} \prec_p \cup \prec_{msg} \right)^*$ where
  - $i \prec_p j$ if and only if $i < j$, for every $i, j \in E_p$
  - $i \prec_{msg} j$ if for some $(p, q) \in Ch$ and $m \in C$ we have:

$$\ell(i) = !(p, q, m) \text{ and } \ell(j) = ?(q, p, m) \text{ and } \sum_{m \in C} |a_1 \ldots a_{i-1}|!(p,q,m) = \sum_{m \in C} |a_1 \ldots a_{j-1}|?(q,p,m)$$

Example

Construct $M(w)$ for $w = !(r, q, m)!!(p, q, m_1)!!(p, q, m_2)?!(q, p, m_1)?!(q, p, m_2)?!(q, r, m)$
Example

Construct $M(w)$ for $w = ! (r, q, m)! (p, q, m_1)! (p, q, m_2)? (q, p, m_1)? (q, p, m_2)? (q, r, m)$

$$E = \{ e_1, \ldots, e_b \}$$

$\leq_r : \emptyset$

$\leq_p : e_2 < p e_3$

$\leq_q : e_4 < q e_5$

$e_5 < q e_b$

$\leq = (\leq_r \cup \leq_p \cup \leq_q \cup \leq_{msg})^+$

$M(w)$
From linearizations to MSCs

For every well-formed word $w \in \text{Act}^*$, $M(w)$ is an MSC.

Relating well-formed words to MSCs

Example:

$M(w_0)$ is not well-formed

$w_0 = !(p,q,a)$

$M(w_1)$ is not well-formed

$w_1 = !(p,q,a) \cdot !(p,q,b) \cdot ?(q,p,b) \cdot ?(q,p,a)$

$M(w_1)$ is not FIFO, not MSC
Proof: for well-formed $w$, $M(w)$ is an MSC (sketch). Let $w$ be $a_1 \ldots a_n$ well-formed.

Construct $M(w)$ by a pass from left-to-right through $w$. Let $w_k = a_1 \ldots a_k$. Start with $w_0 = \epsilon$, the empty word. Take $M(w)$ is empty labeled poset.

Now consider $w_{k+1} \neq \epsilon$ and distinguish 2 cases:

1. $w_{k+1} = w_k ! (p, q, a)$. Then extend $M(w_k)$ with a new event $e_{k+1}$ with $\ell(e_{k+1}) = !(p, q, a)$.

   Extend means that all $e_i \in w_k \cap E_p$ precede $e_{k+1}$ and that $m(e_{k+1})$ is undefined.

2. $w_{k+1} = w_k ? (p, q, a)$. As $w$ is well-formed, $w_k$ is proper (by definition) thus $\exists a_i \in w_k$ with $a_i = !(q, p, a)$ for which $e_i \notin \text{dom}(m)$ in $M(w_k)$. Take the minimal $j$ in $\{1, \ldots, k\}$ with $e_j \notin \text{dom}(m)$. Extend $M(w_k)$ with $e_{k+1}$, $\ell(e_{k+1}) = ?(p, q, m)$ and $m(e_j) = e_{k+1}$. 


(3) As \( u \) is well-timed, it follows that for \( k=n \), the function \( m \) is total on the set of send actions/events in \( W \).
From linearizations to MSCs

Definition

$(E, \preceq, \ell)$ and $(E', \preceq', \ell')$ are isomorphic if there exists a bijection $f : E \rightarrow E'$ such that $e \preceq e'$ iff $f(e) \preceq' f(e')$ and $\ell(e) = \ell'(f(e))$.

Linearizations yield isomorphic MSCs

For every well-formed $w \in Act^*$ and $w' \in Lin(M(w))$: $M(w)$ and $M(w')$ are isomorphic.

$M(w)$ and $M(w')$ are equal except for event identities.
Main theorem

There is a one-to-one relationship between MSC $M$ and the set $\text{Lin}(M)$. 