

1 Lecture 4: Message Sequence Graphs

Theoretical Foundations of the UML

Lecture 4: Message Sequence Graphs

Joost-Pieter Katoen

Lehrstuhl für Informatik 2
Software Modeling and Verification Group

moves.rwth-aachen.de/teaching/ss-16/theoretical-foundations-of-the-uml/

26. Oktober 2017

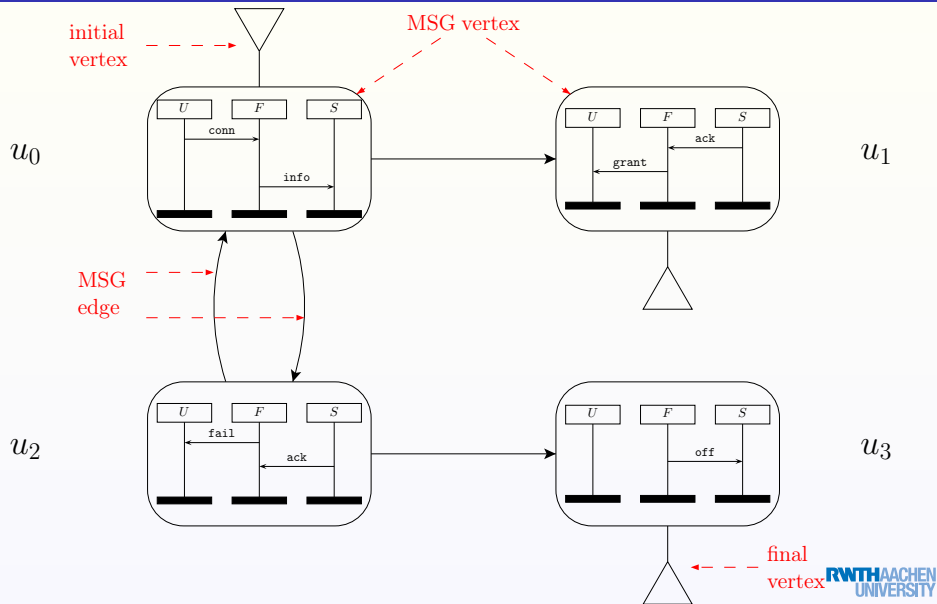
Summary of Lecture #3

- 1 A Message Sequence Chart is a **visual** partial order
 - between send and receive events
 - totally ordered per process vertical ordering
 - receive events happen after their send events horizontal ordering
 - respecting the FIFO property
- 2 **Race**: in practice, the order of receive events cannot be guaranteed
- 3 **Causal order**
 - send events should happen before their matching receive events
 - the ordering of events wrt. sends on same process is respected
 - receive events on a process sent from the same process are ordered as their sends
- 4 A MSC has a **race** if causal order \neq visual order
 - checking whether an MSC has a race can be done in **quadratic** time (in number of events)
 - using an optimized version of **Warshall's** algorithm

The need for composing MSCs

- An MSC describes a possible **single** scenario
 - Typically: a set of scenarios
 - and dependencies between these scenarios:
 - after scenario 1, scenario 2 occurs
 - after scenario 1, scenario 2 **or** 3 occurs
 - scenario 1 occurs **repeatedly**
 - Need for: **sequential composition** (= concatenation),
alternative composition, and
iteration of MSCs
- ⇒ This yields **Message Sequence Graphs**
- Alternatives: ensembles of MSCs, high-level MSCs (**MSC'96**)

Message Sequence Graphs



Message Sequence Graphs

Let \mathbb{M} be the set of MSCs (up to isomorphism, i.e., event identities).

Definition

A **Message Sequence Graph** (MSG) $G = (V, \rightarrow, v_0, F, \lambda)$ with:

- (V, \rightarrow) is a digraph with finite set V of vertices and $\rightarrow \subseteq V \times V$ a set of edges
- $v_0 \in V$ is the starting (or: initial) vertex
- $F \subseteq V$ is a set of final vertices
- $\lambda : V \rightarrow \mathbb{M}$ associates to each vertex $v \in V$, an MSC $\lambda(v)$

Note:

An MSG can be considered as a non-deterministic finite-state automaton without input alphabet where states are MSCs. Obviously, every MSC is an MSG.

Example

Concatenation of MSCs: definition

Let $M_i = (\mathcal{P}_i, E_i, \mathcal{C}_i, l_i, m_i, \preceq_i)$ with $i \in \{1, 2\}$
be two MSCs with $E_1 \cap E_2 = \emptyset$

The **concatenation** of M_1 and M_2 is the MSC
 $M_1 \bullet M_2 = (\mathcal{P}, E, \mathcal{C}, l, m, \preceq)$ with:

$$\begin{aligned} \mathcal{P} &= \mathcal{P}_1 \cup \mathcal{P}_2 & E &= E_1 \cup E_2 & \mathcal{C} &= \mathcal{C}_1 \cup \mathcal{C}_2 \\ & & & \text{(with } E_{?} = E_{1,?} \cup E_{2,?} \text{ etc.)} & & \end{aligned}$$

$$l(e) = \begin{cases} l_1(e) & \text{if } e \in E_1 \\ l_2(e) & \text{if } e \in E_2 \end{cases} \quad m(e) = \begin{cases} m_1(e) & \text{if } e \in E_1 \\ m_2(e) & \text{if } e \in E_2 \end{cases}$$

$$\preceq = (\preceq_1 \cup \preceq_2 \cup \{(e, e') \mid \exists p \in \mathcal{P}. e \in E_1 \cap E_p, e' \in E_2 \cap E_p\})^*$$

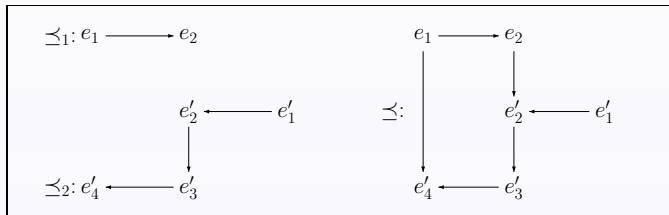
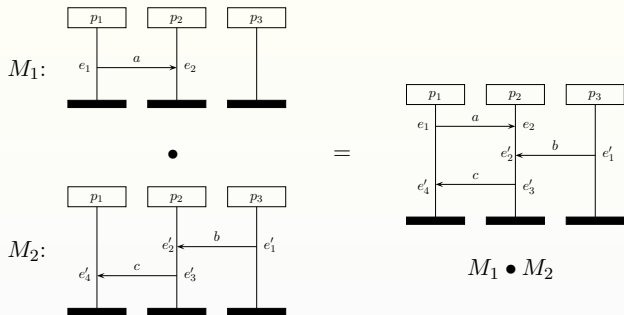
Ordering

$$\preceq = (\preceq_1 \cup \preceq_2 \cup \{(e, e') \mid \exists p \in \mathcal{P}. e \in E_1 \cap E_p, e' \in E_2 \cap E_p\})^*$$

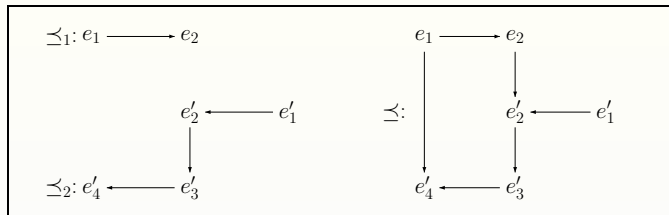
Observations

- events are ordered per **process**:
 - every event at p in MSC M_1 precedes every event at p in MSC M_2
- events at **distinct** processes in M_1 and M_2 can be **incomparable**
- thus: a process may start with M_2 before other processes do pause
- this **differs** from: first complete M_1 , then start with M_2

Example (1)



Example (2)



Note:

Events e_1 and e'_1 are not ordered in $M_1 \bullet M_2$

Example linearizations:

$e_1 \ e_2 \ e'_1 \ e'_2 \ \dots \in \text{Lin}(M_1 \bullet M_2)$

$e'_1 \ e_1 \ e_2 \ e'_2 \ \dots \in \text{Lin}(M_1 \bullet M_2)$

- 1 Concatenation is **associative**:

$$(M_1 \bullet M_2) \bullet M_3 = M_1 \bullet (M_2 \bullet M_3)$$

- 2 Concatenation preserves the **FIFO** property:

$$(M_1 \text{ is FIFO} \wedge M_2 \text{ is FIFO}) \text{ implies } M_1 \bullet M_2 \text{ is FIFO}$$

- 3 Race-freeness, however, is not preserved

$$(M_1 \text{ is race-free} \wedge M_2 \text{ is race-free}) \not\Rightarrow M_1 \bullet M_2 \text{ is race-free}$$

Let $G = (V, \rightarrow, v_0, F, \lambda)$ be an MSG.

A **path** through MSG G is a finite traversal through the graph G .

Definition

A **path** π in MSG G is a finite sequence

$$\pi = u_0 u_1 \dots u_n \text{ with } u_i \in V \text{ (} 0 \leq i \leq n \text{) and } u_i \rightarrow u_{i+1} \text{ (} 0 \leq i < n \text{)}$$

An **accepting** path through MSG G is a path starting in the initial vertex and ending in a final vertex.

Definition

Path $\pi = u_0 \dots u_n$ is **accepting** if: $u_0 = v_0$ and $u_n \in F$.

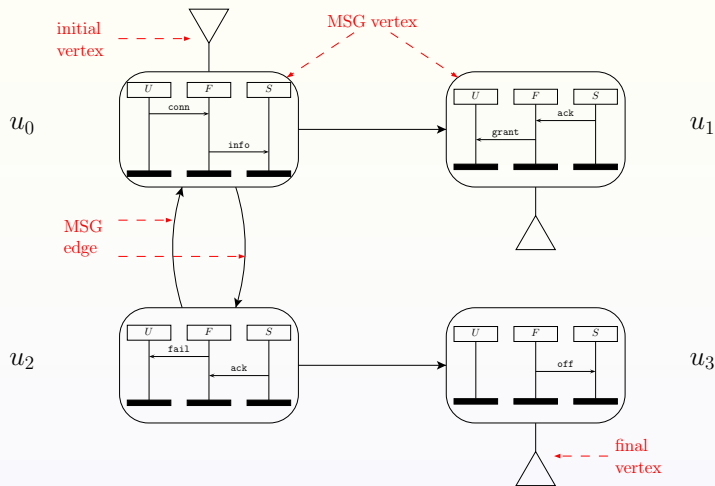
Let $G = (V, \rightarrow, v_0, F, \lambda)$ be an MSG.

Definition

The **MSC of a path** $\pi = u_0 \dots u_n$ through MSG G is defined by:

$$M(\pi) = \underbrace{\lambda(u_0)}_{\text{MSC of } u_0} \bullet \underbrace{\lambda(u_1)}_{\text{MSC of } u_1} \bullet \dots \bullet \underbrace{\lambda(u_n)}_{\text{MSC of } u_n}$$

Example paths



$u_0 u_2 u_0 u_1$ is accepting; $u_0 u_2 u_0 u_2$ is not accepting

Language of an MSG

The **language** of an MSG, i.e., the set of MSCs it represents, is the set of MSCs of its accepting paths.

Definition

The **MSC language** of MSG G is defined by:

$$L(G) = \{M(\pi) \mid \pi \text{ is an accepting path of } G\}.$$

Definition

The **word language** of MSG G is defined by $Lin(L(G))$ where

$$Lin(\{M_1, \dots, M_k\}) = \bigcup_{i=1}^k Lin(M_i).$$

Example

Recall: MSC M has a race if $\preceq \not\subseteq \ll^*$

or, equivalently $Lin(M, \preceq) \not\subseteq Lin(M, \ll^*)$

or, equivalently $Lin(M, \ll^*) \subset Lin(M, \preceq)$

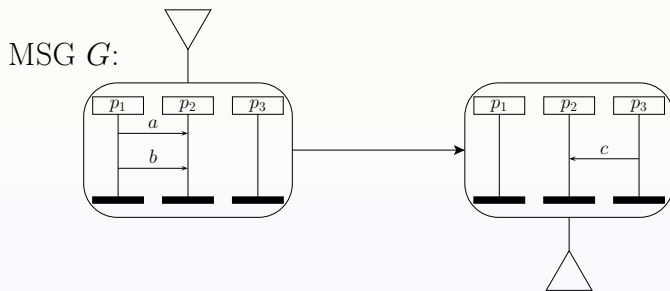
Definition

MSG G has a **race** if $Lin(G, \ll^*) \subset Lin(G, \preceq)$

Example

Definition

MSG G has a race if $Lin(G, \ll^*) \subset Lin(G, \preceq)$



MSG G has a race.

Deciding whether an MSG has a race is undecidable

Theorem

[Muscholl & Peled, 1999]

The decision problem “does MSG G have a race?” is **undecidable**.

Proof.

By a reduction from the universality of semi-trace languages. Requires some Mazurkiewicz’ trace theory. Omitted here. We will see other reduction proofs later on. □

No undecidable problem can ever be solved by a computer or computer program of any kind.

Do MSGs have an MSC in common?

Theorem: undecidability of empty intersection

The decision problem:

for MSGs G_1 and G_2 , do we have $L(G_1) \cap L(G_2) = \emptyset$?

is **undecidable**.

Proof: Reduction from Post's Correspondence Problem (PCP)

... black board ...