Overview

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

http://moves.rwth-aachen.de/teaching/ws-1415/uml/

21. Oktober 2014

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

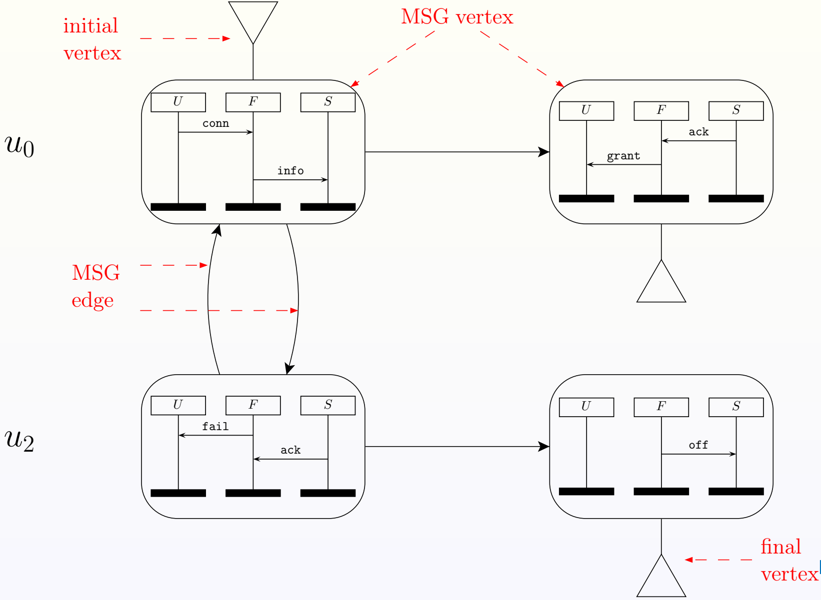
The need for composing MSCs

- 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 $E_1 \cap E_2 = \varnothing$

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:

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

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

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

Concatenation of MSCs: observations

Ordering

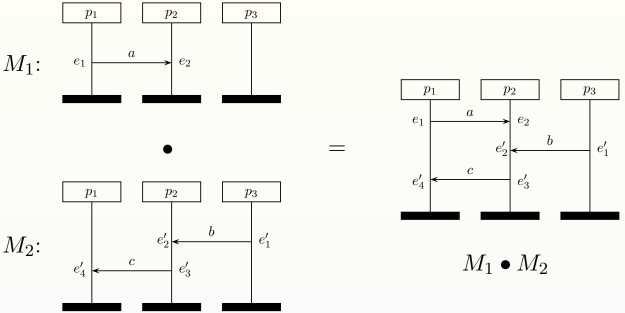
$\preceq \;=\; \big(\preceq_1 \cup \preceq_2 \cup \{(e,e') \mid \exists p \in \mathcal{P}.\, e \in E_1 \cap E_p\,,\; e' \in E_2 \cap E_p\}\,\big)^*$

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

RWTH AACHEN UNIVERSITY



$$
\begin{tikzcd}
\preceq_1\colon e_1 \arrow[r] & e_2 & \\
 & e'_2 \arrow[d] & e'_1 \arrow[l] \\
\preceq_2\colon e'_4 & e'_3 \arrow[l] &
\end{tikzcd}
\qquad
\preceq\colon
\begin{tikzcd}
e_1 \arrow[r] \arrow[dd] & e_2 \arrow[d] & \\
 & e'_2 \arrow[d] & e'_1 \arrow[l] \\
e'_4 & e'_3 \arrow[l] &
\end{tikzcd}
$$

Example (2)

$$
\begin{tikzcd}
\preceq_1: e_1 \arrow[r] & e_2 & \\
& e'_2 \arrow[d] & e'_1 \arrow[l] \\
\preceq_2: e'_4 & e'_3 \arrow[l] &
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
& e_1 \arrow[r] \arrow[dd] & e_2 \arrow[d] & \\
\preceq: & & e'_2 \arrow[d] & e'_1 \arrow[l] \\
& e'_4 & e'_3 \arrow[l] &
\end{tikzcd}
$$

Note:

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

Example linearizations:

$e_1 \quad e_2 \quad e'_1 \quad e'_2 \quad \ldots \in Lin(M_1 \bullet M_2)$

$e'_1 \quad e_1 \quad e_2 \quad e'_2 \quad \ldots \in Lin(M_1 \bullet M_2)$

Properties of concatenation

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:

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

3. Race-freeness, however, is not preserved

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

Paths in MSGs

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

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

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

Definition

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

Paths in an MSG represent MSCs

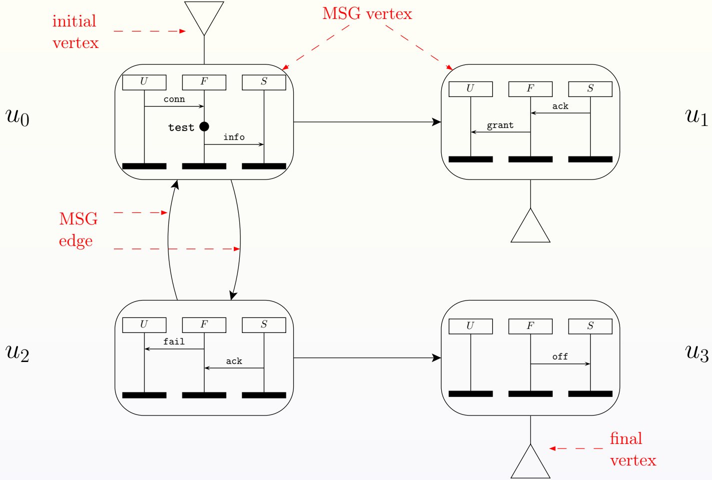
Let $G = (V, \rightarrow, v_0, F, \lambda)$ be an MSG.

Definition

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

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

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

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

Example

Races in MSGs

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

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

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

Definition

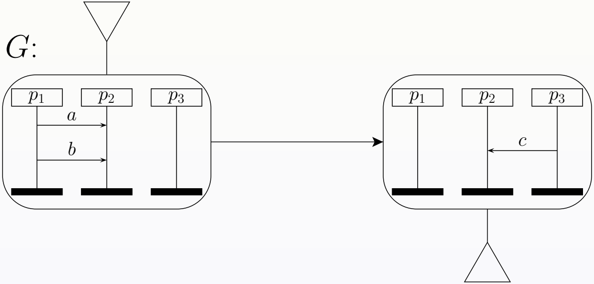
MSG G has a race if $Lin(G, \ll^*) \subset Lin(G, \preceq)$

Example

Definition

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

MSG G:



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. $\square$

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

The state space of an MSC

State of an MSC

Let MSC M with event set E. The set $E' \subseteq E$ is a state of the MSC M whenever for all $e \in E'$ it holds $e' \preceq e$ implies $e' \in E'$, i.e., E' is downward-closed wrt. $\preceq$.

The set of states of MSC M is called M's state space. Every MSC has a finite state space.

State space of an MSG

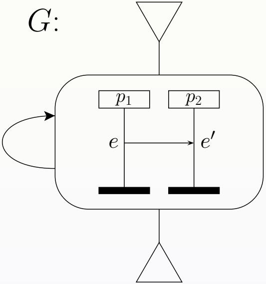
The state space of MSG G is the union of the state spaces of M_i for all $M_i \in L(G)$.

Expressiveness of MSGs (1)

Observation 1:

The state space of an MSG G may be infinite.

G:



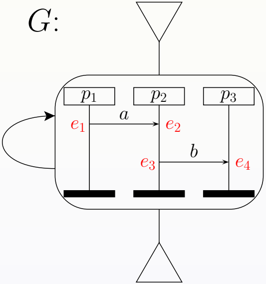
A possible state is $\{e^{(1)}, e^{(2)}, e^{(3)}, \ldots\}$
(where $e^{(i)}$ is the occurrence of e in the i-th iteration)

$\Longrightarrow$ system that realizes G requires unbounded communication channel

Expressiveness of MSGs (2)

Observation 2:

The linearizations of an MSG may not be context-free.



Linearizations of MSG G are of the form $\{e_1^k\ e_2^l\ e_3^m\ e_4^n \mid k \geqslant l \geqslant m \geqslant n\}$.

This language is not context-free.

Permuting the order of events

Let $w, w' \in E^*$, and M an MSC with event set E. Then it holds:

(1) $w\ e\ e'\ w' \in Lin(M)$, $l(e) = ?(q,p,b)$
$l(e') = !(p,q,a)$

implies $w\ e'\ e\ w' \in Lin(M)$. not the reverse!

(2) $w\ e\ e'\ w' \in Lin(M)$, $l(e) = !(p,q,a)$ and
$l(e') = ?(q,p,b)$

$$\underbrace{\sum_{m \in \mathcal{C}} |w|_{!(p,q,m)}}_{\substack{\text{number of sends} \\ \text{from } p \text{ to } q \text{ in } w}} > \underbrace{\sum_{m \in \mathcal{C}} |w|_{?(q,p,m)}}_{\substack{\text{number of receipts} \\ \text{of } q \text{ from } p \text{ in } w}}$$

implies $w\ e'\ e\ w' \in Lin(M)$.

(3) $w\ e\ e'\ w' \in Lin(M)$, $e \in E_p$, $e' \in E_q$, $p \neq q$

and e, e' do not match like in (1) or (2)

implies $w\ e'\ e\ w' \in Lin(M)$.

Expressiveness of MSGs (4)

Observation 3:

The set of linearizations of an MSG is context-sensitive.

Note:

Rule (2) is a context-sensitive rule of form $X\, a\, b\, Y \longrightarrow X\, b\, a\, Y$ as its applicability depends on the number of sends and receipts in the context X.

Note:

The results so far do not imply that any context-sensitive language is MSG-definable.

Context sensitivity (informal argument)

- Take MSG G and use vertex identities as vertex labels.
- $K(G) =$ set of "accepting" vertex sequences. This is regular.
- Replace each vertex v by $Lin(\lambda(v))$
  (interpret sequencing element wise)
- Let the resulting set be $\widetilde{K}(G)$. This is regular.
- Close $\widetilde{K}(G)$ under the permutation rules (1), (2), (3)
  (cf. previous two slides)

The resulting word language is context-sensitive.

Do MSGs have an MSC in common?

Theorem: undecidability of empty intersection

The decision problem:

$$\text{for MSGs } G_1 \text{ and } G_2, \text{ do we have } L(G_1) \cap L(G_2) = \varnothing?$$

is undecidable.

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

. . . black board . . .