Foundations of the UML

Lecture 3: Message Sequence Graphs

Joost-Pieter Katoen

Lehrstuhl für Informatik 2
Software Modeling and Verification Group

http://moves.rwth-aachen.de/i2/370

26. Oktober 2009

Message Sequence Chart (MSC) (1)

Definition

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

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

$$E = \biguplus_{p \in \mathcal{P}} E_p = \underbrace{E_? \cup E_! \cup E_{\text{loc}}}_{\text{partitioning of } E}$$

- $\mathcal{C}$, a finite set of message content
- $l : E \to Act$, a labelling function defined by:

$$l(e) = \begin{cases} !(p,q,a) & \text{if} \quad e \in E_p \cap E_! \\ ?(p,q,a) & \text{if} \quad e \in E_p \cap E_? \\ p(a) & \text{if} \quad e \in E_p \cap E_{\text{loc}} \end{cases} \quad , p \neq q \in \mathcal{P},\ a \in \mathcal{C}$$

UNIVERSITY

Message Sequence Chart (MSC) (2)

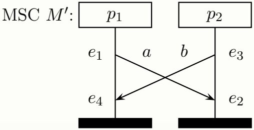
Definition

- $m : E_! \to E_?$ a bijection ("matching function"), satisfying:

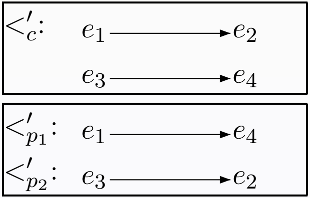
$$m(e) = e' \wedge l(e) = !(p, q, a) \text{ implies } l(e') = ?(q, p, a) \;\; (p \neq q, \;\; a \in \mathcal{C})$$

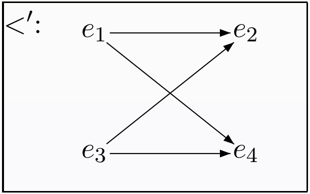
- $< \subseteq E \times E$ is a partial order ("visual order")

$$< = \left(\bigcup_{p \in \mathcal{P}} <_p \;\cup\; \{(e, m(e)) \mid e \in E_!\} \right)^*$$

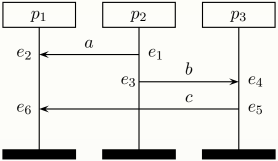


$$M' = (\underbrace{\mathcal{P}, E, \mathcal{C}, l, m}_{\text{as above}}, <') \text{ with:}$$





Visual order vs. possible order



$e_2 < e_6$?

If message b takes much shorter than message a,
then c might arrive at p_1 before a!

Formally: $<_{p_1} = \{e_6, e_2\}$ is possible but $\neq$ visual order.

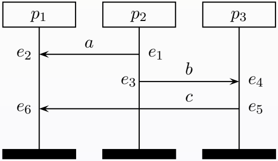
When are such situations possible and how to detect them?

Races (1)

- Let $M = (\mathcal{P}, E, \mathcal{C}, l, m, <)$ be an MSC.
- Let $\ll \; \subseteq E \times E$ be defined by:

$$
\begin{array}{lll}
e \ll e' & \text{iff} & e' = m(e) \\
& \text{or} & e <_p e' \text{ and } E_! \cap \{e, e'\} \neq \varnothing \\
& \text{or} & e, e' \in E_p \cap E_? \text{ and } m^{-1}(e) <_q m^{-1}(e')
\end{array}
$$

$\ll$ is the "interpreted / possible order" (also called causal order)



Example

$e_1 \ll e_2, \quad e_3 \ll e_4, \quad e_5 \ll e_6, \quad e_1 \ll e_3, \quad e_4 \ll e_5, \quad \neg(e_2 \ll e_6)$

Races (2)

Definition

MSC M contains a race if for some $e, e' \in E_?$:

$$e <_p e' \text{ but } \neg(e \ll^* e')$$

where $\ll^* \subseteq E \times E$ is the reflexive and transitive closure of $\ll$.

- How to check whether MSC M has a race?

  compute $\ll^*$ and compare to $<_p$

- $\ll^*$ can be computed using Floyd-Warshall's algorithm

  worst-case time complexity $\mathcal{O}(|E|^3)$, improved here to $\mathcal{O}(|E|^2)$

Races (3)

MSC M has a race if $< \not\subseteq \ll^*$ or equivalently:

$$\exists e, e' \in E_? \, . \, (e <_p e' \text{ and } e \not\ll^* e')$$

$\Rightarrow$ protocol implementation based on $<_p$ may cause problems, e.g.,

1. unspecified message reception
2. deadlock situations
3. use content of wrong message

Computing $\ll^*$: Warshall's algorithm

Algorithm

$\underbrace{\text{compute } \ll^*}_{\text{Warshall's Algorithm}}$ and compare with $<$

Warshall's Algorithm: <u>input:</u> $R \subseteq X \times X$ where X is a set
<u>output:</u> R^*

Idea:

Consider R and R^* as directed graphs

There is an edge $x \Rightarrow y$ in R^* iff there is a (possibly empty) path

$$x = x_0 \rightarrow x_1 \rightarrow x_2 \rightarrow \ldots \rightarrow x_n = y \text{ in } R$$

(our setting: $X = E, R = \ll, \ R^* = \ll^*$)

Warshall's algorithm

- assume: vertices are numbered $\{1, 2, \ldots, n\}$ where $n = |E|$
- for $j \in \{1, \ldots, n{+}1\}$ define relation $\overset{j}{\Longrightarrow}$ as follows:
  $x \overset{j}{\Longrightarrow} y$ iff $\exists$ path in R from x to y such that all vertices on the path ($\neq x, y$) have a smaller number than j
- Then:
  (1) $x \Longrightarrow y$ iff $x \overset{n+1}{\Longrightarrow} y$
  (2) $x \overset{1}{\Longrightarrow} y$ iff $x = y$
  (3) $x \overset{y+1}{\Longrightarrow} z$ iff $x \overset{y}{\Longrightarrow} z$ or $x \overset{y}{\Longrightarrow} y \overset{y}{\Longrightarrow} z$
- Algorithm: determine the relations $\overset{1}{\Longrightarrow}, \ldots, \overset{n}{\Longrightarrow}, \overset{n+1}{\Longrightarrow}$ iteratively using properties (1) + (2)
- Store $\overset{i}{\Longrightarrow}$ in a boolean matrix C
- Postcondition: $C[x, y] = \texttt{true}$ iff $(x, y) \in R^*$
- Precondition: $\forall x, y \in X \;.\; C[x, y] = \texttt{false}$

Warshall's algorithm (1)

for $x := 1$ to n do
    for $y := 1$ to n do
        $C[x, y] := (x = y \text{ or } \underbrace{(x, y) \in R}_{x \ll y})$

/* loop invariant */
/* after the j-th iteration of outermost loop it holds: $C[x, y]$ iff $x \overset{j+1}{\Longrightarrow} y$ */
for $y := 1$ to n do
    for $x := 1$ to n do
        if $C[x, y]$ then
            for $z := 1$ to n do
                if $C[y, z]$ then
                    $C[x, z] :=$ `true`

Correctness and complexity

Lemma: correctness

After j iterations: $x \overset{j+1}{\Longrightarrow} y$ iff $C[x, y] = 1$.

Proof:

by induction on j; on the black board

Complexity

Time complexity of Warshall's algorithm : $\mathcal{O}(n^3)$ where $n = |X|$

Proof:

follows from the fact that each loop has at most n iterations.

Efficiency improvement [Alur et al. '96] (1)

Warshall's algorithm determines R^* for any binary relation R.

Using some properties of $\ll$ the complexity can be improved.

Exploit that for $\ll$:

- $\ll$ is acyclic
- number of immediate predecessors of $e \in E$ under $\ll$ is at most two (why?)
- Note: e is an immediate predecessor of e' if:

$$e \ll e' \text{ and } \neg(\exists e'' \notin \{e, e'\}.\ e \ll e'' \ll e')$$

Efficiency improvement [Alur et al. '96] (2)

Body of the algorithm for detecting races now becomes:

for $e := 1$ to n do
  for $e' := e - 1$ downto 1 do
    if $C[e', e]$ then
      for $e'' := 1$ to $e' - 1$ do
        if $C[e'', e']$ then
          $C[e'', e] := \texttt{true}$

this part is executed for (e, e') only if e' is an immediate predecessor of e, i.e., # inner loops $\leq 2 \cdot n \Longrightarrow$ time complexity $\mathcal{O}(n^2)$

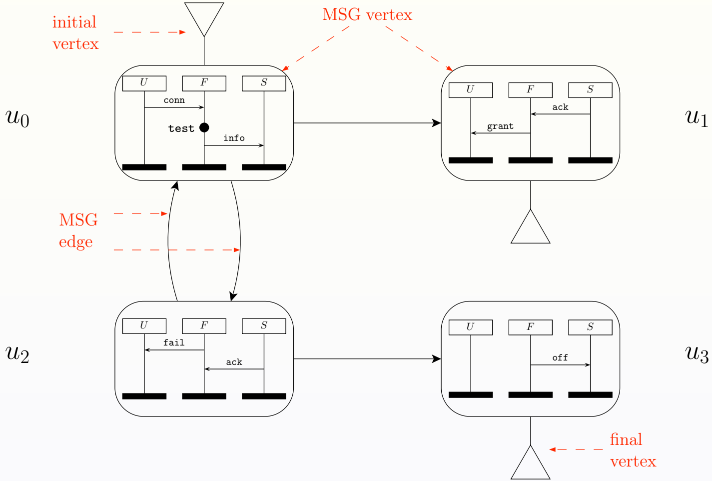
Sets of MSCs

- MSC specifies a single scenario
- Typically: a set of scenarios
- + an ordering relation between them:
  - 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



$$u_0 \; u_2 \; u_0 \; u_1 = \lambda(u_0) \bullet \lambda(u_2) \bullet \lambda(u_0) \bullet \lambda(u_1)$$

Definition

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

A Message Sequence Graph (MSG) G is a tuple $G = (V, \rightarrow, v_0, F, \lambda)$ 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:

1. an MSG is an NFA without input alphabet where states are MSCs
2. every MSC is an MSG

Concatenation of MSCs (1)

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

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

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

$$< \, = \, <_1 \cup <_2 \cup \{(e, e') \mid \exists p \in \mathcal{P}. \, e \in E_1 \cap E_p \, , \; e' \in E_2 \cap E_p\}$$

RWTH AACHEN UNIVERSITY

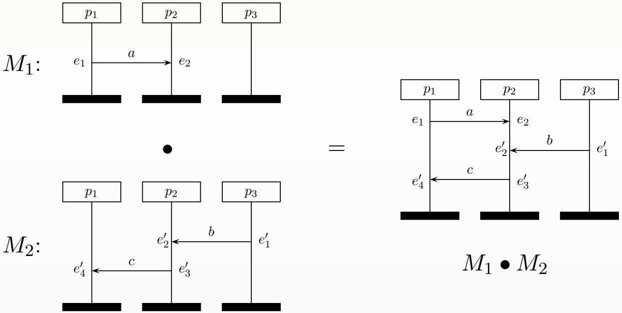
Concatenation of MSCs (2)

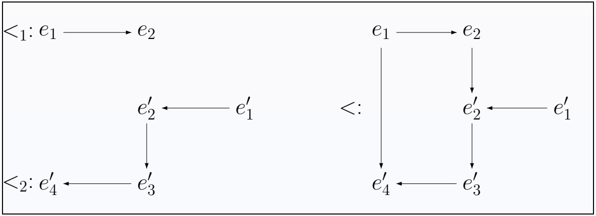
Note

- events are ordered process-wise:

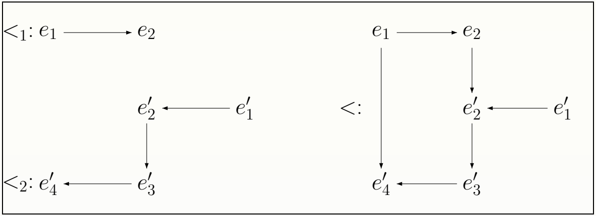
  $$\textit{events at } p \textit{ in MSC } M_1 \textit{ precede events at } p \textit{ in MSC } M_2$$

- thus: some processes may proceed to M_2 before others!
- $\neq$: first complete M_1 then execute M_2


M1:
p1
p2
p3
e1
a
e2
•
=
M2:
e'2
b
e'1
e'4
c
e'3
M1 • M2


<1: e1
e2
e'2
e'1
<2: e'4
e'3
<:
e1
e2
e'2
e'1
e'4
e'3

Example (2)



Note:

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

Example:

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

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

Properties

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:

$$M_1 \text{ is FIFO} \;\;\&\;\; M_2 \text{ is FIFO} \quad \text{implies} \quad M_1 \bullet M_2 \text{ is FIFO}$$

3. Race-freeness, however, is not preserved

$$M_1 \text{ is race-free} \;\;\&\;\; M_2 \text{ is race-free} \quad \not\Rightarrow \quad M_1 \bullet M_2 \text{ is race-free}$$

RWTH AACHEN UNIVERSITY

Preliminaries

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

Definition

A path π of G is a finite sequence

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

Definition

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

Definition

The MSC of a path $\pi = u_0 \; \ldots \; u_n$ is:

$$M(\pi) \;=\; \underbrace{\lambda(u_0)}_{\text{MSC of } u_0} \bullet \underbrace{\lambda(u_1)}_{\text{MSC of } u_1} \bullet \ldots \bullet \underbrace{\lambda(u_n)}_{\text{MSC of } u_n} \;=\; \prod_{i=0}^{n} \lambda(u_i)$$

Language of an MSG

Definition

The (MSC) language of MSG G is defined by:

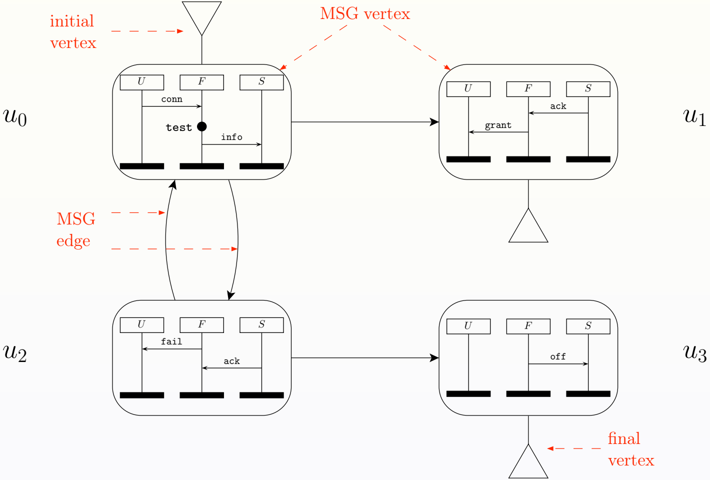
$$L(G) = \{M(\pi) \mid \pi \text{ is an accepting path of } G\}.$$

Definition

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

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

Example



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

Races in MSGs

Recall: MSC M has a race if $< \not\subseteq \ll^*$
or, equivalently $Lin(E,<) \not\subseteq Lin(E,\ll^*)$
or, equivalently $Lin(E,<) \subset Lin(E,\ll^*)$

Definition

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

Theorem ([Musch† & Peled '99])

The decision problem "MSG G has a race" is undecidable.

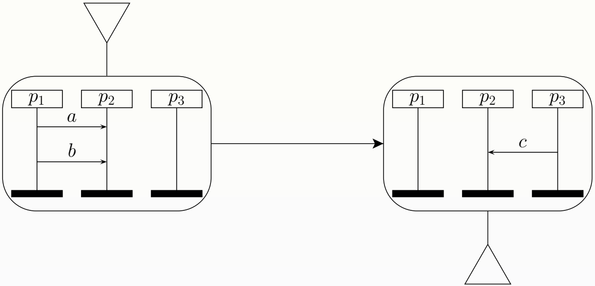
Proof.

by a reduction from Post's Correspondence Problem (PCP). Not easy.
We will see a similar—though simpler—proof later on. □

RWTH AACHEN UNIVERSITY

Example

G:



MSG G has a race.

Expressiveness of MSGs (1)

Fact 1:

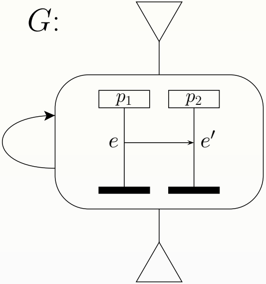
The state space of an MSGs may be infinite.

The state of an MSC with event set E is $E' \subseteq E$ such that $e \in E' \wedge e' < e \Longrightarrow e' \in E'$ (i.e., E' is downward-closed wrt. $<$)

The set of states of M is M's state space

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

Example



G is infinite state

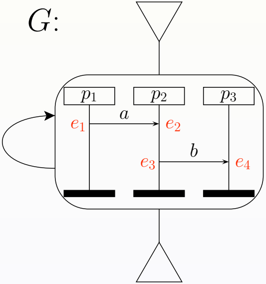
A possible state is $\{e^{(1)}, e^{(2)}, e^{(3)}, \ldots\}$
(where $e^{(i)}$ is the occurrence of e in the i-th iteration)

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

Expressiveness of MSGs (1)

Fact 2:

The state space of an MSG may not be context-free.



States of G are of the form $\{e_1^k\, e_2^l\, e_3^m\, e_4^n \mid k \geq l \geq m \geq n\}$

This language is not context-free

Expressiveness of MSGs (1)

Fact 3:

The state space of an MSG is context-sensitive.

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

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

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

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

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

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

RWTH AACHEN UNIVERSITY

Expressiveness of MSGs (2)

(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) (cf. previous slide)

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

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
- Replace each vertex v by $Lin(\lambda(v))$
  (interpret sequencing element wise)
- Let the resulting set be $\widetilde{K}(G)$
- Close $\widetilde{K}(G)$ under the permutation rules (1), (2), (3)
  (cf. previous slides)

The resulting word language is context-sensitive.