FOUNDATIONS OF THE UML

Lecture 2:

Message Sequence Graphs

Joost-Pieter Katoen

MOVES

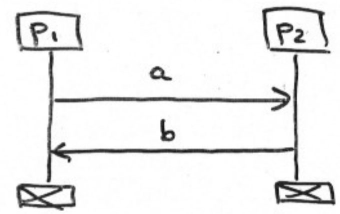
moves.rwth-aachen.de/i2/192

WS 2007/2008

November 5, 2007

MESSAGE SEQUENCE CHARTS

- Scenario-based language
- Primary use: requirements specification
- Visual Language



- standardized by the ITU
- Adopted by the UML (sequence diagrams)
- Widely used in industrial practice
- "Easy" to comprehend ?

DEFINITION

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

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

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

- $\mathbb{C}$, finite set of message content
- $\ell : E \rightarrow Act$, a labelling function

$$\ell(e) = \begin{cases} p!q(a) & \text{if } e \in E_p \cap E_! \\ p?q(a) & \text{if } e \in E_p \cap E_? \\ p(a) & \text{if } e \in E_p \cap E_{loc} \end{cases} \quad \begin{matrix} p \neq q \\ a \in \mathbb{C} \end{matrix}$$

- $m : E_! \rightarrow E_?$ a bijection ("matching function")

satisfying: $m(e) = e' \wedge \ell(e) = p!q(a)$
$(p \neq q, a \in \mathbb{C})$ implies $\ell(e') = q?p(a)$

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

$$< = \underbrace{\bigcup_{p \in P} <_p}_{\substack{<_p \text{ is total order} \\ = \text{"top-to-bottom" order} \\ \text{on process } p}} \cup \underbrace{\{(e, m(e)) \mid e \in E_!\}}_{\substack{\text{communication order} \\ <_c}}$$

FIFO Property

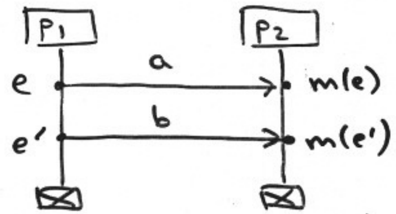
MSC $M = (\mathcal{P}, E, \mathbb{C}, \ell, m, <)$ has the First-Out (FIFO) property whenever:

for all $e, e' \in E_!$ we have

$$e < e' \wedge \ell(e) = p!q(a) \wedge \ell(e') = p!q(b)$$

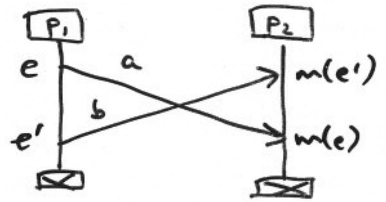
implies $m(e) < m(e')$

"no message overtaking"



FIFO

$\ell(e) = p_1!p_2(a)$
$\ell(e') = p_1!p_2(b)$
$e < e'$
$\Longrightarrow m(e) < m(e')$



non-FIFO

Note: we assume an MSC to possess the FIFO property, unless stated otherwise

LINEARIZATIONS

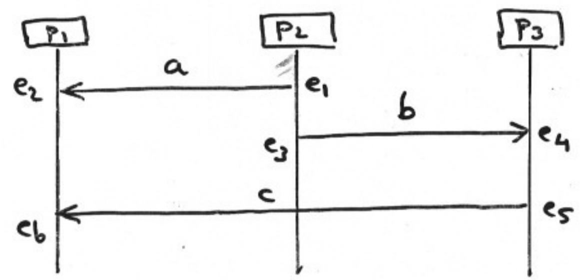
Lin (M) = set of linearizations of MSC M

<u>Lemma:</u> there is a one-to-one correspondence between MSCs and their sets of linearizations

so: Lin (M) uniquely characterizes M !

VISUAL ORDER VS. POSSIBLE ORDER

causal order



$e_2 < e_6$?

if b takes much shorter than a,

then c might arrive at P_1 before a !

formally: $<_{P_1} = \{ (e_6, e_2) \}$ is possible

$\neq$ visual order

when are such situations possible?

RACES

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

$e \ll e'$ iff $e' = m(e)$

at least e or e' is a send

or $e <_p e'$ and either e or $e' \in E_!$

or $e, e' \in E_p \cap E_?$ and $m^{-1}(e) <_q m^{-1}(e')$

$\ll$ is the "interpreted / possible order"

- <u>Example</u> (cf. previous slide)

$e_1 \ll e_2$, $e_3 \ll e_4$, $e_5 \ll e_6$, $e_1 \ll e_3$, $\underline{e_4 \ll e_5}$

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

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

reflexive and transitive closure of $\ll$

- How to check whether MSC M has a race?

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

(Floyd-Warshall $\mathcal{O}(|E|^3)$, but here $\mathcal{O}(|E|^2)$).

COMPUTING $\ll^*$: WARSHALL'S ALGORITHM

MSC M has a <u>race</u> 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. unspecified reception, deadlock, or use information from wrong message

<u>Algorithm</u>: $\underbrace{\text{compute } \ll^*}_{\text{Warshall's algorithm}}$ and compare with $<$

Warshall's algorithm: <u>input</u>: $R \subseteq X \times X$

<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

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

(our case: $R = \ll$, $R^* = \ll^*$)

WARSHALL'S ALGORITHM

- assume: vertices are numbered $\{1, 2, \ldots, n\}$
- for $j \in \{1, \ldots, n\}$ define relation $\overset{j}{\Rightarrow}$ as follows:

  $x \overset{j}{\Rightarrow} y$ <u>iff</u> $\exists$ 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) $x \Rightarrow y$ iff $x \overset{n+1}{\Rightarrow} y$

  (2) $x \overset{1}{\Rightarrow} y$ iff $x = y$

  (3) $x \overset{y+1}{\Rightarrow} z$ iff $x \overset{y}{\Rightarrow} z$ or $x \overset{y}{\Rightarrow} \underline{\underline{y}} \overset{y}{\Rightarrow} z$

- Algorithm: determine the relations $\overset{1}{\Rightarrow}, \overset{2}{\Rightarrow}, \ldots, \overset{n}{\Rightarrow}, \overset{n+1}{\Rightarrow}$ iteratively using properties (1) + (2)
- Store $\overset{i}{\Rightarrow}$ in a boolean matrix C
- Post condition : $C[x,y] = \text{true}$ iff $(x,y) \in R^*$
- Pre condition: $\forall x, y \in X . \; C[x,y] = \text{false}$

WARSHALL'S ALGORITHM

```
for x:=1 to n do          /* initialization */
    for y:=1 to n do
        C[x,y] := (x=y or (x,y) ∈ R)
od od                        \_____/
                              x<<y

/* loop invariant:                          */
/* after the j-th iteration of outermost    */
/* loop it holds: C[x,y] iff x =j+1=> 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 fi
            od
        fi
od od
```

Correctness: after j iterations: $x \overset{j+1}{\Longrightarrow} y$ iff C[x,y]

Proof: by induction on j

Time complexity: $O(n^3)$ where $n = |X|$

EFFICIENCY IMPROVEMENT

[Aluret. al. '96]

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:

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

Body of the algorithm 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]:=true
                                           fi
        od
    fi
od od
```

this part is executed for (e,e′) only if e′ is an immediate predecessor of e, i.e.

innerloops $\leq 2 \cdot n$

time complexity $O(n^2)$ $\Leftarrow$

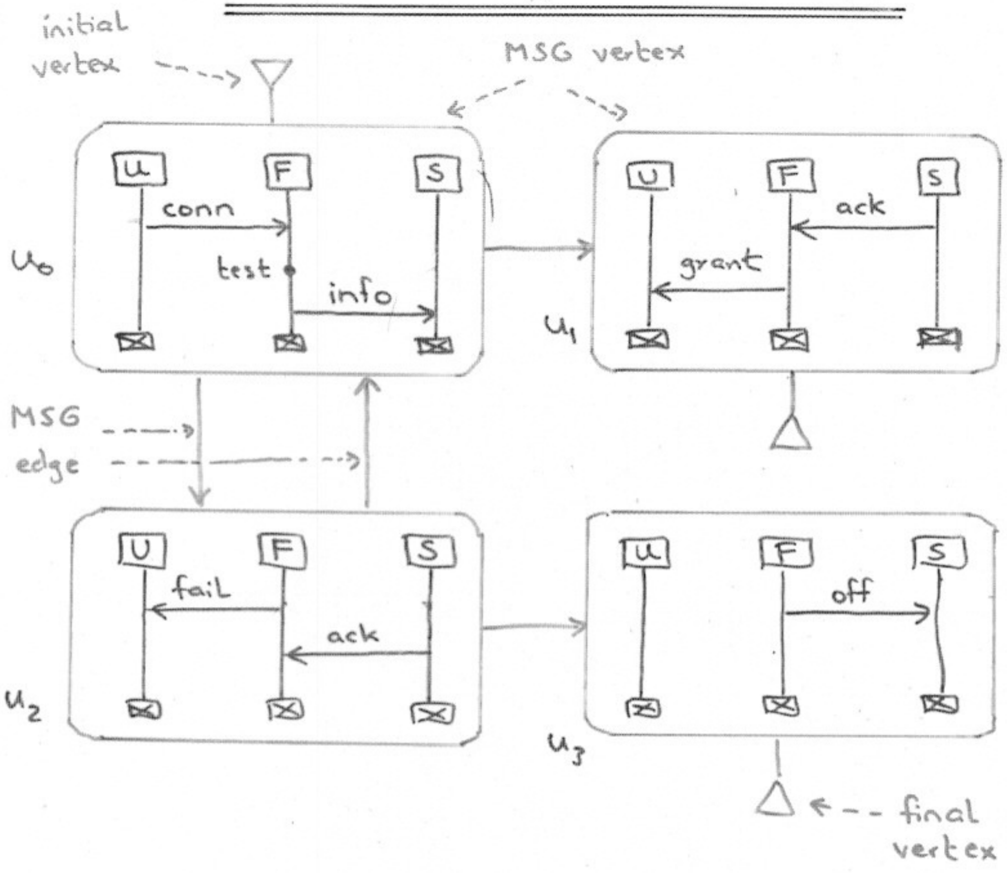
SETS OF MSCs

- MSC specifies a <u>single</u> scenario
- Typically : a <u>set</u> 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: alternative composition,
  sequential composition, (= concatenation)
  iteration of MSCs

This yields $\boxed{\text{Message Sequence Graphs}}$

- Alternatives: ensembles of MSCs,
  high-level MSCs (MSC'96)

MESSAGE SEQUENCE GRAPHS



$$u_0\, u_2\, u_0\, u_1 \;=\; \lambda(u_0)\cdot\lambda(u_2)\cdot\lambda(u_0)\cdot\lambda(u_1)$$

DEFINITION

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

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

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

CONCATENATION OF MSCs

Let $M_i = (P_i, E_i, \mathbb{C}_i, \ell_i, m_i, <_i)$ $\quad i \in \{1,2\}$

be an MSC with $E_1 \cap E_2 = \emptyset$

The concatenation of M_1 and M_2 is the MSC

$$M_1 \bullet M_2 = (P, E, \mathbb{C}, \ell, m, <) \quad \text{with:}$$

$$P = P_1 \cup P_2 \qquad E = E_1 \cup E_2 \qquad \mathbb{C} = \mathbb{C}_1 \cup \mathbb{C}_2$$

(with $E_? = E_{1,?} \cup E_{2,?}$ etc.)

$$\ell(e) = \begin{cases} \ell_1(e) & \text{if } e \in E_1 \\ \ell_2(e) & \text{if } e \in E_2 \end{cases} \qquad m(e) = \begin{cases} m_1(e) & \text{if } e \in E_1 \\ m_2(e) & \text{if } e \in E_2 \end{cases}$$

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

same process

Note:

- events are ordered process-wise

  events at p in M_1 precede events at p in M_2

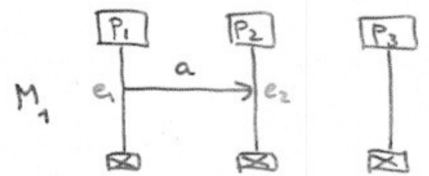
- thus: some processes may proceed to M_2 before others!

  see Exercises

- $\neq$: first complete M_1 then execute M_2

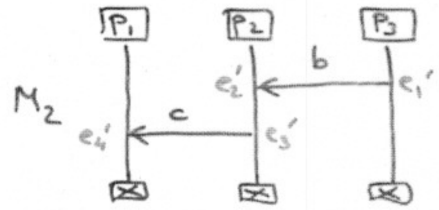
EXAMPLE

M_1

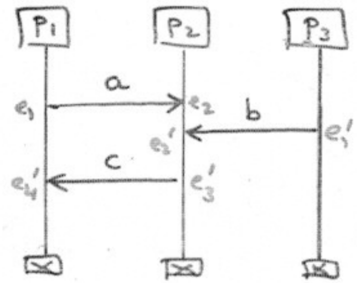


$\bullet$

M_2



$=$



$M_1 \bullet M_2$

$<_1$: $e_1 \longrightarrow e_2$

$<_2$:

$$\begin{tikzcd} e_2' \arrow[d] & e_1' \arrow[l] \\ e_3' & \\ \end{tikzcd}$$

$$e_4' \longleftarrow e_3'$$

$<$:

$$\begin{tikzcd} e_1 \arrow[r] \arrow[dd] & e_2 \arrow[d] & \\ & e_2' \arrow[d] & e_1' \arrow[l] \\ e_4' & e_3' \arrow[l] & \end{tikzcd}$$

Note: e_1 and e_1' are not ordered in $M_1 \cdot M_2$

e.g. $e_1\, e_2\, e_1'\, e_2' \ldots \in Lin(M_1 \cdot M_2)$

$e_1'\, e_1\, e_2\, e_2' \ldots \in Lin(M_1 \cdot M_2)$

LANGUAGE OF AN MSG

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

A path π of G is a finite sequence

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

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

$$M(\pi) = \lambda(u_0) \cdot \lambda(u_1) \cdot \ldots \cdot \lambda(u_n)$$
$$= \prod_{i=0}^{n} \lambda(u_i)$$

MSC concatenation

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

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

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

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

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

RACES IN MSGs

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

or, equivalently $Lin(E,<) \not\subseteq Lin(E, \ll^*)$

or, equivalently $Lin(E,<) \subset Lin(E, \ll^*)$

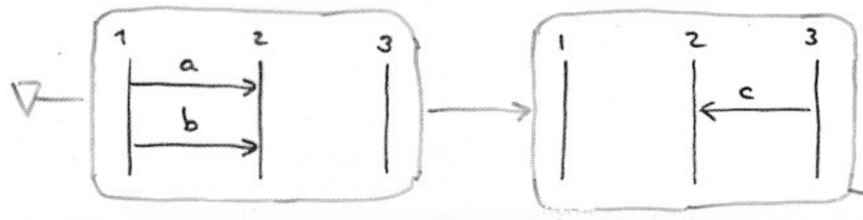
(since $Lin(E,<) \subseteq Lin(E, \ll^*)$)

MS<u>G</u> G has a <u>race</u> if $Lin(G,<) \subset Lin(G, \ll^*)$

<u>Theorem</u> (without proof) [Musholl & Peled '99]

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

<u>Proof</u>: by a reduction from Post's Correspondence Problem (PCP). Not easy. We will see a similar — though simpler — proof later on.



has a race

EXPRESSIVENESS OF MSGs

Fact 1: MSGs may represent infinite-state systems

Def: the state of an MSC with event set E is

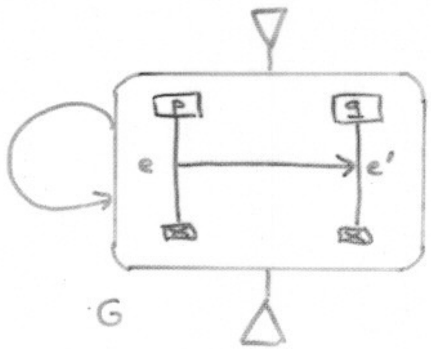
$E' \subseteq E$ such that $e \in E' \wedge e' < e \rightarrow e' \in E'$

(i.e.: E' is downward-closed wrt. $<$)

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

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

Example



G is infinite state

possible state is

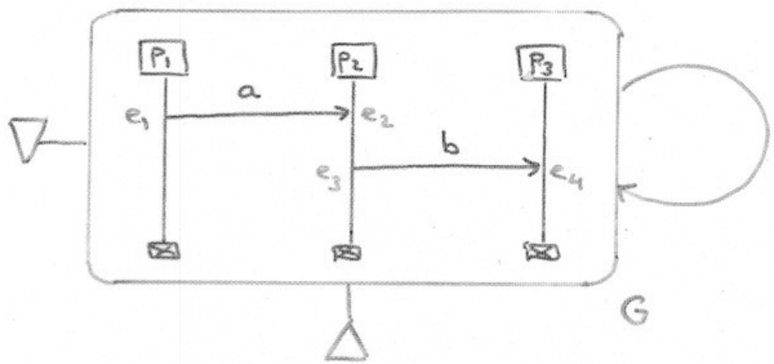
$\{e^{(1)}, e^{(2)}, e^{(3)}, \ldots\ldots\}$

$e^{(i)}$ is occurrence of e in i-th iteration

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

EXPRESSIVENESS OF MSGs

Fact 2: state space of MSG may not be context-free



states of G are of the form

$$\{e_1^k\, e_2^\ell\, e_3^m\, e_4^n \mid k \geq \ell \geq m \geq n\}$$

This language is not context-free

EXPRESSIVENESS OF MSGs

> Fact 3 : the state space of an MSG is context-sensitive

(1) let $w, w' \in E^*$, and M an MSC with event set E

$w\,\underline{e}\,\underline{e'}\,w' \in Lin(M)$, $\ell(\underline{e}) = q?p(b)$, $\ell(\underline{e'}) = p!q(a)$

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

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

$$\underbrace{\sum_{a \in \mathbb{C}} |\underline{\underline{w}}|_{p!q(a)}}_{\text{number of sends from } p \text{ to } q \text{ in } w} > \underbrace{\sum_{a \in \mathbb{C}} |\underline{\underline{w}}|_{q?p(a)}}_{\text{number of receipts of } q \text{ from } p \text{ in } w}$$

implies $w\,\underline{e'}\,\underline{e}\,w' \in Lin(M)$.

(3) $w\,\underline{e}\,\underline{e'}\,w' \in Lin(M)$, $\underline{e} \in E_p$, $\underline{e'} \in E_q$, $p \neq q$ and $\underline{e}, \underline{e'}$ do not match like in (1) + (2)

implies $w\,\underline{e'}\,\underline{e}\,w' \in Lin(M)$

Note : rule (2) is a <u>context-sensitive</u> rule of form $X\,\underline{a}\,\underline{b}\,Y \longrightarrow X\,\underline{b}\,\underline{a}\,Y$

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 slide)

The resulting language is context sensitive

INTERSECTION OF MSGs

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 ......